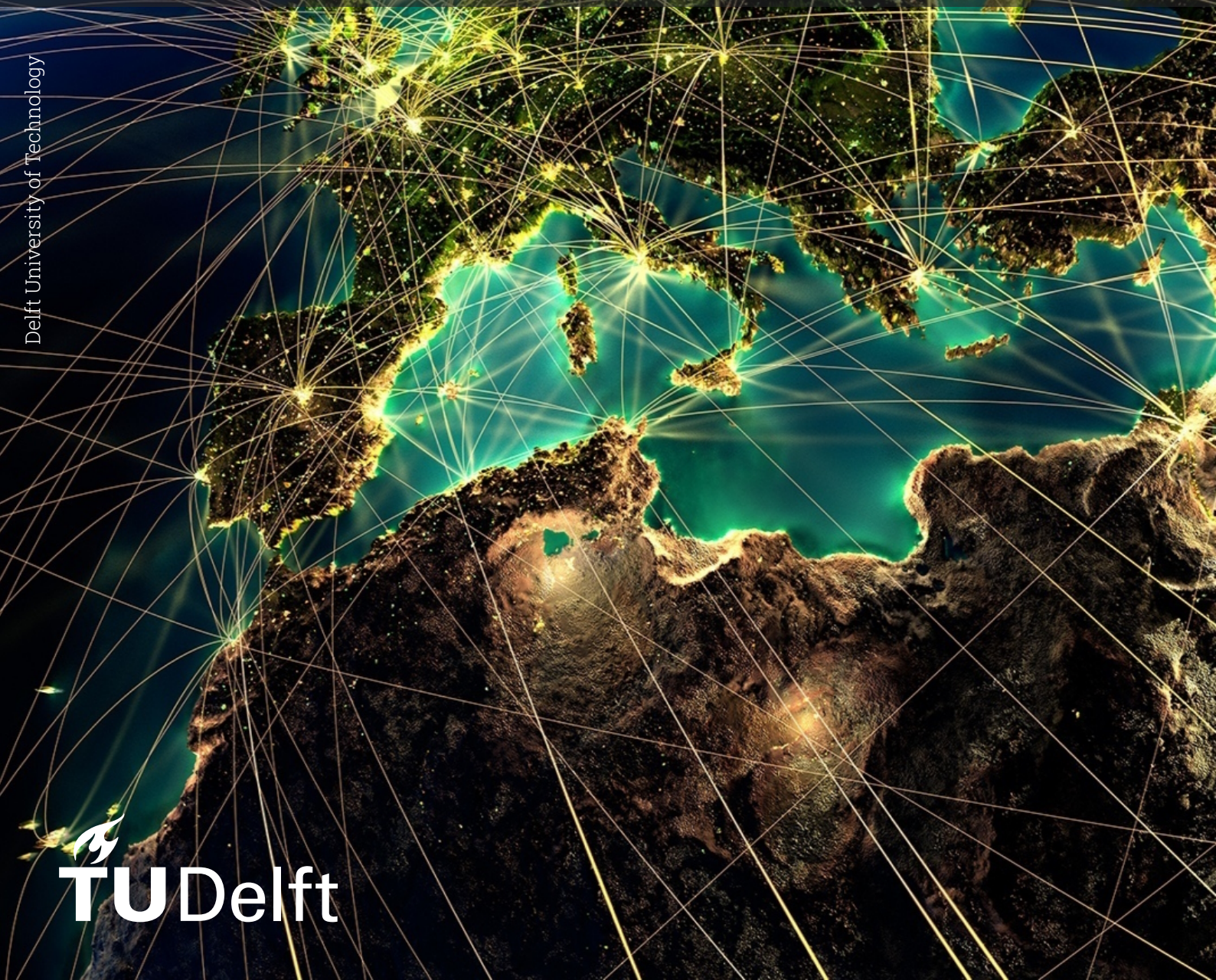
Machine learning based trajectory prediction to support demand forecasting

AE5310: Thesis Control and Operations

R.A. Vos

Delft University of Technology



TU Delft

Machine learning based trajectory prediction to support demand forecasting

by

R.A. Vos

In fulfilment for the degree of Master of Science
at the Delft University of Technology

Student number:	4588304	
Date:	October 23th 2023	
Project Duration:	October 2022 - November 2023	
Supervisors:	Dr. J. Sun	TU Delft
	Prof. Dr. Ir. J.M. Hoekstra	TU Delft
	F. Dijkstra,	KDC

TU Delft

Preface

The document in front of you is my final thesis report that concludes the Master's program at the Aerospace Engineering faculty of Delft University of Technology. This thesis is conducted at the Air Traffic Management group from the Control and Operations Master track. This report includes the final academic paper, as well as the preliminary report from the first half of the project.

First and foremost, I want to sincerely thank Junzi Sun, Ferdinand Dijkstra and Jacco Hoekstra for supervising this project. It was a great pleasure to work with you. Your enthusiasm for aviation and air traffic management has motivated me to get my teeth into the problem. Moreover, your critical attitude, sharp questions and knowledgeable insights have been a valuable contribution to the project. I am proud of the results and hope it can be a worthy contribution to your academic work as well. Furthermore, I would like to thank KDC and LVNL for their support of this project. Without the provided data, hardware and expertise, these results would not have been achievable. Specifically, I want to thank Evert Westerveld for the opportunity to be involved with KDC and to graduate from the Centre of Excellence.

With this report, I conclude my time as a student at TU Delft, which in the end could not have been successful without the support of my family. I want to thank my parents for always standing behind me. But most importantly I want to thank my wife Sanne for sharing the highs and lows and for your infinite support through both the exciting and tough moments. It would not have been possible without you.

Reinier Vos
Delft, October 2023

Contents

List of Figures

List of Tables

List of Abbreviations

Part I

Scientific Article

Machine learning based trajectory prediction to support air traffic demand forecasting

A Transformer Neural Network Approach

R.A. Vos

Under supervision of Dr. J. Sun, Prof. Dr. Ir. J.M. Hoekstra & F. Dijkstra

Control and Operations, Faculty of Aerospace Engineering
Delft University of Technology, Delft, The Netherlands

Air traffic sector demand and capacity balancing is an important process to enable safe and efficient flight execution. In current operations, demand and capacity are determined based on schedules and flight plans. In reality, disruptions to flights create a different situation that may not have been anticipated by the Air Navigation Service Provider (ANSP). Wrong demand forecasts may cause unnecessary network regulations or inefficient flight execution. This research aims to improve air traffic sector demand forecasting, by exploring machine learning-based trajectory prediction. In light of the Trajectory Based Operations (TBO) concept that is being developed within Air Traffic Management (ATM) research, a trajectory-based approach is taken to improve demand forecasts. To achieve this, the transformer neural network was identified as a suitable generative model that can predict aircraft trajectories. Using available traffic messages from the Eurocontrol Business-to-Business (B2B) connection, and actual trajectories obtained from the OpenSky ADS-B repository, a successful transformer neural network was built. This trajectory predictor could accurately generate trajectories, outperforming the flight plan and other neural network approaches by a large margin. For demand prediction, the introduction of improved trajectories provided small gains that could potentially lead to more stable predictions.

ATM | Demand forecasting |Trajectory Based Operations | Transformer Neural Network

I. Introduction

With the ever-increasing numbers of flights, ANSPs are experiencing significant challenges to maintain capacity and improve the sustainability performance within their respective airspace. ATM is concerned with managing the air traffic and airspace, such that flights are executed safely and efficiently. This includes services such as air traffic control, navigation, information and emergency services, but also demand and capacity balancing of the different airspace sectors.

Demand and capacity balancing is the process of managing the traffic flows through an airspace block such that safe and efficient flight operation can be guaranteed. Where demand is the number of flights inside the sector, and the capacity is the ability of the responsible ANSP to keep separation between flights and allow throughput. Air Traffic Control The Netherlands (LVNL), the Dutch ANSP, currently makes demand forecasts based on received flight information, amongst which the flight plan is a very important element to estimate the arrival time in the Dutch airspace. However, a lot of uncertainty is present in this data. The research objective is therefore to improve air traffic sector demand forecasting, by exploring machine learning-based trajectory prediction. For tactical decisions from the air traffic controller supervisor, the demand predictions at a three-hour look ahead time are most relevant. With the available information three hours before the arrival of a flight, a machine learning model is applied to generate trajectories that should give a more accurate representation of the flight and as a result, increase the predictability of air traffic demand. This study focuses on the Dutch airspace as the area of interest and is supported by LVNL with data and expertise.

First, in section II, an overview of the related academic advancements in the field will be discussed. Thereafter, section III explains the chosen methodology and experiment setup. In section IV the results of the experiment are provided, after which section V is a discussion on the observed outcome. To end, section VI contains the conclusion of this research paper.

II. Related Work

A. Demand Forecasting. To effectively balance the number of aircraft in an air traffic sector (demand) to the available capacity of the sector, ANSPs make forecasts on the expected traffic numbers in the coming hours. Conventionally, the number of flights in the sector is determined based on flight schedules, or flight plans. In reality, however, flights may deviate from their planning, either through delays, rerouting, or other disruptions. This introduces uncertainty to the demand forecast, which may either cause unanticipated traffic overload or loss of capacity. Könnemann [1] has made an extensive decomposition of all factors influencing demand prediction deficits for an upper area sector. Departure time prediction was found to be the largest source of uncertainty, followed by Air Traffic Control (ATC) interference, and trajectory prediction uncertainty. As a result, numerous research efforts have tried to improve demand predictions. Generally, two methods can be derived from literature: The trajectory-based approach and the aggregate approach.

The trajectory-based approach considers individual flights to

determine when the flight is inside the sector. This method includes conventional demand prediction based on flight plans, but has recently seen new developments. Gilbo & Smith [2] were able to reduce the demand prediction error of US air traffic sectors with a regression model. The model improved the prediction error variance by about 0.5 aircraft, while still relying on flight plans only. Fernández et al.[3] predicted 4D trajectories using a Hidden Markov Model and used these subsequent trajectories to determine sector demand. The advantage of using trajectories is that it gives insight into which flights contribute to the demand. Fernández et al. demonstrated that by imposing delays on the start time of trajectories, demand and capacity can be optimised. Xu, Prats and Delahaye [4] take this one step further and also optimise demand and capacity with variable route options or different sector configurations. The optimisation was proven very effective with minimal imposed delay. This clearly shows that trajectory-based demand prediction is a versatile solution.

Aggregate demand predictions do not rely on individual flights, but consider the entire flow of traffic through the sector. Sridhar et al.[5] and Menon et al.[6] modelled blocks of airspace as control volumes that convey a traffic flow. The result is a linear discrete-time dynamic system, which can potentially be upgraded to an automated control system. Ma et al.[7] created a spatiotemporal graph network to represent traffic flowing through the air traffic sectors in France. Passing flow through the graph network and propagating via the message-passing algorithm allows the prediction of traffic numbers for each sector. Significant improvements have been reached, especially on the 1-2-hour look ahead time compared to other neural networks. Aggregate models generally show better predictive performance in experimental studies, but are not knowingly applied in operational environments today.

B. Trajectory Prediction. Given that this research builds on the TBO framework, the trajectory prediction methodology is fundamental to the success of the research. Two global methods can be distinguished when investigating trajectory prediction. Classical trajectory prediction is based on a mathematical model that physically calculates the future state based on the current state, intent and constraints. At the heart of such models is usually a kinetic or kinematic model that computes the motion of the aircraft according to Newtonian physics. Examples of this can be the OpenAP WARP model created by Sun [8], or the Eurocontrol BADA model. Although very successfully applied in operational environments, a pitfall of model-based Trajectory Predictor (TP) is the required set of constraints and intent information. Especially for longer-term predictions, a lack of right intent information can reduce prediction accuracy drastically. Given the look-ahead time of 3 hours that is chosen in this research, the data-driven trajectory prediction approach is a more suitable option.

In recent academic works, trajectory prediction has shifted towards data-driven methods, relying on the vast amount of data that has become available. Especially since the introduction of Automatic Dependent Surveillance - Broadcast (ADS-B) equipment, aircraft surveillance data has become more accessible. Furthermore, machine learning methodology has developed significantly, showing impressive advances in various fields of research. In trajectory prediction, recurrent neural networks are some of the latest applications. Overkamp [9] for example, has shown that Long-Short Term Memory Cell (LSTM) networks can accurately predict trajectories in free-route airspace up to 20 minutes look ahead time. In his research, the versatility of the network was shown by including both spatiotemporal traffic density data as well as temporal trajectory information. Liu & Hansen [10] proved that LSTM networks can also be applied to generatively predict trajectories on longer look-ahead times. Using the last filed flight plan and encoded spatiotemporal weather data, the model predicts entire trajectories for a single city pair. With average horizontal errors of around 50NM and vertical errors of 2800ft, the predictive performance of the model is a reasonable improvement over the flight plan accuracy. When making demand forecasts, however, flights are not limited to a single city pair. Rozendaal [11] showed that a bi-directional LSTM network may have the potential to accurately predict trajectories for flights towards Schiphol airport.

The previous section shows that LSTM neural networks have been extensively tested in the trajectory prediction domain, but are not the only successful data-driven method. For example, clustering is frequently applied to the trajectory prediction problem. Various studies use clustering to pre-group trajectories and train a consecutive predictor. Examples of this are the methods proposed by Fernández et al.[3], and Wu et al.[12]. Fernández et al. clustered flight plans with the K-NN method based on distance and weather metrics. After having grouped the flight plans into 5-8 clusters, the dimensionality is reduced significantly. A Hidden Markov Model was then applied to each cluster, allowing for better convergence. Wu et al. used a similar approach, but used a K-medoid clustering approach and convolutional-LSTM neural network. In both cases, the predicted trajectory outperformed similar methods without clustering the input.

C. Machine learning. Although various machine learning models have been applied to trajectory and demand prediction, new methods have been found in other academic domains, which can be relevant to this research. One of the latest developments is the transformer neural network as proposed by Vaswani et al.[13]. Similar to recurrent neural networks, this model was originally designed for language translation tasks. Using the self-attention mechanism, computations could be parallelised. Classical recurrent networks such as the LSTM network rely on sequential calculations, which are much more resource-intensive. The encoder-decoder transformer model performs similar or better on the English-German and English-French translation

tasks with significantly lower training costs. This breakthrough model resulted in many variations of the model such as BERT & GPT, which have shown outstanding results in various natural language processing tasks. Moreover, biological modelling advancements have also been conceived by transformer neural networks as surveyed by Zhang et. al.[14]. During this research, no direct applications of the transformer neural network were found in the domains of aircraft trajectory prediction, but Wang et al.[15] and Achaji et al.[16] did apply a transformer model to road traffic and pedestrian trajectory prediction respectively. Furthermore, the attention mechanism has been exploited in part for traffic flow forecasting. Ma et al.[7] have successfully implemented the attention mechanism in a graph neural network for demand prediction of the French airspace sectors.

Although the performance of the transformer neural network is groundbreaking, varieties of the attention mechanism are being explored to reduce computational load even further. A recent finding in this field is centred around token mixing algorithms. Because of the scaled dot product, the attention mechanism complexity increases quadratically with the length of the input sequence. Token mixing algorithms find substitutes for this operation. Lee-Thorp et al.[17] for example, introduced a Fast Fourier Transformation layer to convert the input signal to a frequency vector. This is a very fast operation, yet allows the network to find internal correlations and dependencies. Alternatively Tolstikhin et al.[18] applied a multilayer perceptron instead of attention. This type of feed-forward neural network also mixes the signal tokens, without losing information. Both methods show performance on par with current state-of-the-art neural networks. Lee-Thorp et al. showed that their model performed almost as well as BERT on text classification tasks, yet computed a sample more than twice as fast. This makes token-mixing algorithms an interesting path of development.

III. Methodology

The methodology of this research is to create demand forecasts via a trajectory prediction approach. Based on related academic work, and the available data, a transformer neural network is proposed to generate trajectories. First, the available data and processing steps are discussed. Thereafter, the proposed trajectory prediction models are explained. In the fourth section, the training process is explained. Consecutively the demand forecasting model is introduced, after which the experiment setup is presented in the final section.

A. Input data. Because this research is relevant to operational decision-making, the selected input data must be readily available in the operational domain. For this reason, the following data sources were consulted to construct the input and testing datasets:

1) *Eurocontrol B2B flight messages*: The Eurocontrol B2B connection sends flight status messages to LVNL containing information such as a flight plan, departure status, origin, destination, aircraft registration, and airline. In addition, it may also include timings amongst which the Estimated Off Block Time (EOBT), taxi time & Estimated Time of Arrival (ETA). An anonymised data excerpt can be found in Appendix A. This data is currently used to make demand forecasts for the coming 3-5 hours and has a high level of integrity. This makes it a suitable starting point for the model input. For this research, flight messages from May 2021 are available. This is during the COVID period, which means lower traffic figures that could result in different flight behaviour. Nonetheless, it is expected that this dataset is still suitable for assessing the capabilities of the model.

2) *OpenSky ADS-B trajectories*: To train a model to predict trajectories, the actual flown trajectories must be specified. Because LVNL does not have surveillance data for complete origin-to-destination flights, this data must be sought elsewhere. ADS-B surveillance data is tracked and stored by several contributors on the collaborative OpenSky network. This research makes thankful use of this data that spans many parts of the globe. This data includes frequent 4D position reports, but also includes aircraft state parameters such as origin, destination, aircraft id, speed and heading. Although not all flights are present in the dataset, and not every trajectory is complete, this dataset is still very suitable for training the trajectory predictor model.

3) *ERA5 meteorological data*: Various studies have shown the importance of including meteorological conditions in trajectory prediction models [10]. Hence, this will be used as part of the input dataset. The European Centre for Medium-Range Weather Forecasts (ECMWF) historic database provides meteorological parameters such as wind, temperature and precipitation on pressure levels from 1000-1hPa. The spatial resolution of this data is 31km, and spans globally. For this research, the atmospheric parameters are taken at timesteps 0:00, 06:00, 12:00, and 18:00 UTC, for the pressure levels from 1000hPa to 125hPa. The northerly and easterly wind components are taken, as well as the temperatures, as it is expected that these values influence the trajectory most significantly. Convective weather is also considered an important contributor, but this is left out because of computing limitations.

4) *Airspace data*: To make demand forecasts, trajectories must be overlaid with the relevant airspace block, to test when and where the airspace is crossed. The airspace layout is taken from the Eurocontrol Network Manager (NM) Demand data repository.

B. Data processing. Before using the data in the model, it must first be prepared and merged into a format that is suitable for machine learning purposes. This research makes use of the PyTorch machine learning library, hence the final dataset must be provided in the tensor format. To achieve this, first, all flights are resampled to a temporal resolution of 4 minutes and are zero-padded to a fixed length of 220

samples. Secondly, the weather conditions along the flight path are added to the B2B flight messages. Consecutively, flight messages must be matched to the ADS-B trajectory, to train the model with the right target trajectory. This is done based on the corresponding flight number and departure time. Because of data size and computing resources, this research is limited to a single look-ahead time. As a result, the last available flight message 3 hours before the actual arrival is taken as model input. The third step is to format the data, which will be explained in more detail below. Finally, the data is split into a training and test dataset. The test dataset consists of 10 full days of traffic randomly sampled from the entire dataset. Complete days are required, because not only trajectory prediction performance is evaluated, but also a complete demand forecast must be analysed. Demand forecasting error can only be determined using a bulk of flights, which required a larger portion (30%) of the available dataset than a conventional 80-20 data split. As a result of the smaller remaining dataset, it was decided that this is used entirely for model training. This means that no validation dataset can be used to evaluate performance during training. Alternatively, training is done in steps with intermediate tests to capture potential overfitting.

Providing data in the right format is an important aspect of machine learning. Because mathematical models do not understand the physical meaning of variables, everything has to be sized and shaped into numbers relatable to the network. Preferably, to keep the network bounded, all input variables are normalised to values between 0-1. This can be done with min-max normalisation, as shown in Equation 1.

$$x_{norm} = \frac{(x - x_{min})}{(x_{max} - x_{min})} \tag{1}$$

Nonetheless, normalising every parameter is not always straightforward. A good example is the coordinate system used to define flight plan and trajectory locations. The spherical coordinate system presents trajectories on a non-linear relational basis and is therefore very challenging to capture by a neural network, as was argued by Overkamp [9] & Tran et al.[19]. Hence, the trajectories are first transferred to another coordinate system. The coordinates of the flight plan and ADS-B trajectory are expressed in terms of latitude (ϕ) and longitude (λ) in degrees and height (h) in feet. This can be converted to the Earth-Centered, Earth-Fixed (ECEF) reference frame using Equation 2 and Equation 3. Where a and b are the equatorial and polar earth radius respectively. Consecutively, these coordinates can be transformed to the East North Up (ENU) reference frame using Equation 4. In this equation, X_r,Y_r & Z_r are the reference location which is set to Amsterdam Airport (ICAO: EHAM). Finally, the flight plan is normalised from the ENU x and y coordinates to a range between 0 (origin) and 1 (Amsterdam Airport Schiphol), using transformation matrix A derived from Equation 5. Note that the transformation is first made to a range between 1 and 2, and then shifted. This is required because a unique inverse matrix of the 0 and 1 range does not exist.

The resultant transformation matrix is then used to normalise the target ADS-B trajectory, and also to convert the output signal back to an actual output trajectory. The advantage of normalising trajectories is that it may eliminate the need for other dimensionality reduction steps such as clustering.

$$\begin{aligned} X_c &= N(\phi) + h cos\phi cos\lambda \\ Y_c &= N(\phi) + h cos\phi sin\lambda \\ Z_c &= N(\phi) + h sin\phi \end{aligned} \tag{2}$$

$$N(\phi) = \frac{a^2}{\sqrt{a^2 cos^2\phi + b^2 sin^2\phi}} \tag{3}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -sin\lambda_r & cos\lambda_r & 0 \\ -sin\phi_r cos\lambda_r & -sin\phi_r sin\lambda_r & cos\phi_r \\ cos\phi_r cos\lambda_r & cos\phi_r sin\lambda_r & sin\phi_r \end{bmatrix} \begin{bmatrix} X_c - X_r \\ Y_c - Y_r \\ Z_c - Z_r \end{bmatrix} \tag{4}$$

$$A = \begin{bmatrix} x_{origin} & y_{origin} \\ x_{EHAM} & y_{EHAM} \end{bmatrix}^{-1} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{5}$$

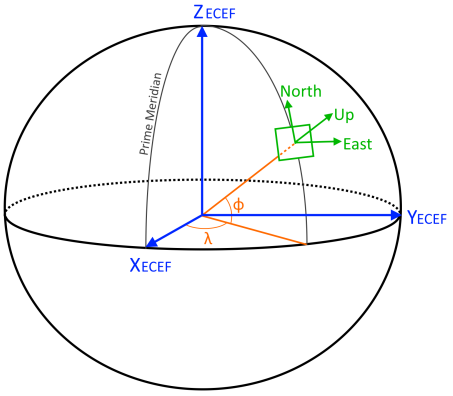


Fig. 1. Reference frames used for geographic positions: Yellow is the spherical coordinate system. Blue is the ECEF reference frame. Green is the Cartesian ENU reference frame.

Besides normalisation, some variables require other data formatting steps. Static, non-time dependent variables such as origin airport, airline operator and aircraft type for example must be encoded numerically. Integer encoding was selected for these variables, representing each unique variable as an integer value. Moreover, temporal static data such as the day of the week, or departure time are cyclically encoded. This technique accounts for the flaws of min-max normalisation in cyclical data. For example, Sunday and Monday are very close time-wise, but after integer encoding, Monday will receive a value of 0 and Sunday will receive a value of 6. For this reason Equation 6 is applied to represent cyclical patterns in the data. Note that it results in two variables, as a single sinusoidal encoding could yield the same result for two values due to symmetry. As a result, the final dataset and performed processing are given in Table 1.

$$\begin{aligned} H_{sin} &= sin(\frac{2\pi H}{max(H)}) \\ H_{cos} &= cos(\frac{2\pi H}{max(H)}) \end{aligned} \tag{6}$$

Table 1. Overview of the input and target tensor variables, including the applied data processing steps that were taken. Note that the target tensor is presented as the difference between the actual trajectory and the flight plan. During training, it was found that this resulted in considerably better results compared to a target that is the actual trajectory.

INPUT TENSOR

Time Series Feature	Processing	Dim
FPL duration timestamp	-	220
FPL latitude	Trajectory norm.	220
FPL longitude	Trajectory norm.	220
FPL altitude	Min-Max norm.	220
North wind component	Min-Max norm.	220
East wind component	Min-Max norm.	220
Air temperature	Min-Max norm.	220

TARGET TENSOR

Time Series Feature	Processing	Dim
ADS-B duration timestamp	-	220
Δ(ADS-B, FPL) latitude	Trajectory norm.	220
Δ(ADS-B, FPL) longitude	Trajectory norm.	220
Δ(ADS-B, FPL) altitude	Min-Max norm.	220

Static Feature	Processing	Dim
Estimated take-off time	Cyclical Encoding	2
Day of the week	Cyclical Encoding	2
Pre-departure delay	Min-Max norm.	1
Origin airport	Integer encoding	1
Aircraft Type	Integer encoding	1
Aircraft Operator	Integer encoding	1

C. Trajectory prediction model. Based on the literature survey, the transformer neural network was selected as the most suitable candidate for the trajectory prediction task. The original transformer neural network was developed by Vaswani et al.[13], and consists of an encoder-decoder structure as shown in Figure 2. The left side of the figure shows the encoder and the right column shows the decoder. For the trajectory prediction task, the original transformer had to be adapted slightly, which will be explained in more detail. Nonetheless, this was kept to a minimum in order to properly evaluate the potential performance of the transformer network. Looking over the elements of the transformer neural network in Figure 2, the encoder input is first processed by a linear input embedding layer. Secondly, a positional embedding is added such that the model can relate between relative positions of input tokens. The positional encoding is sampled from a sinusoidal relation as shown in Equation 7.

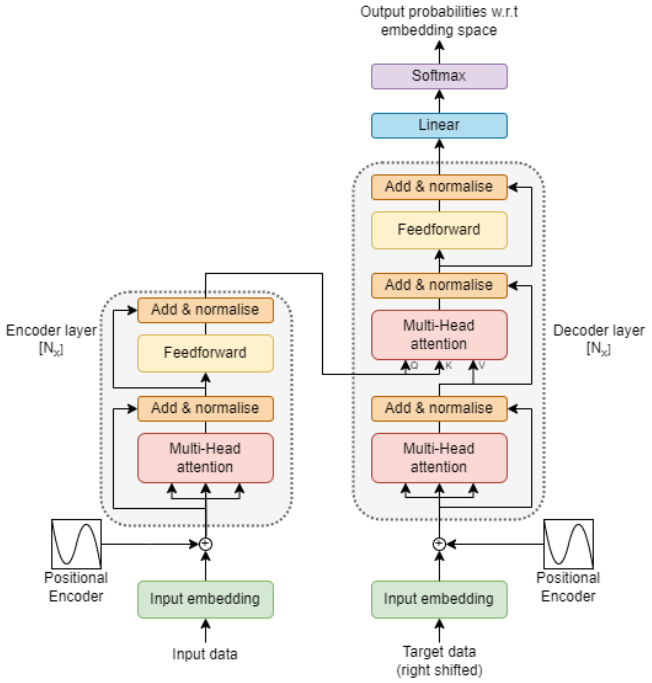


Fig. 2. Schematic overview of the transformer neural network as proposed by Vaswani et al.[13].

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \quad \textbf{(7)}$$

The encoder layer that follows consists of three types of sub-layers: A multi-head attention layer, an addition & normalisation layer, and a feedforward layer. As the scheme shows, the input signal is duplicated to a bypass channel for each functional layer, and added afterwards, which makes sure that the model does not suffer from vanishing or exploding gradient effects. The multi-head attention mechanism is presented in more detail in Figure 3, where (a) shows the attention mechanism, and (b) is the multi-headed attention layer. The input signal is duplicated into the keys (K), queries (Q), and values (V) signal. The keys and queries are parsed through a matrix multiplication layer, after which the outcome is scaled with the signal dimension (d_k), masked (optionally), and taken through a soft-max operation. The dot product of the resultant matrix and the values from the original signal are taken as the final attention outcome. Equation 8 shows the mathematical expression of attention. In multi-headed attention, the attention layer is repeated in parallel h times, where h is the number of heads. This is a network design parameter, which was set to 8 in the original transformer. The number of encoder layers can also be freely determined but was set to 4 in accordance with the original transformer.

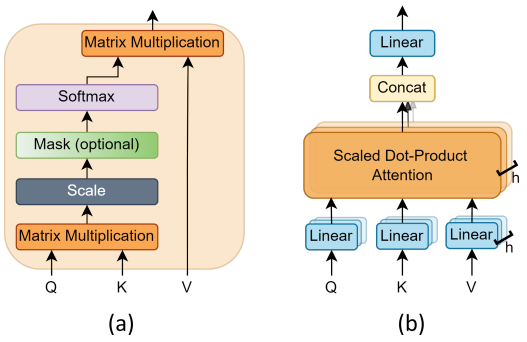


Fig. 3. (a) Schematic representation of the attention mechanism. (b) Multi-headed attention as applied in the original transformer neural network [13].

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad \textbf{(8)}$$

When looking at the decoder layer, various elements are similar to the encoder structure and need no further clarification. However, two primary differences are apparent. The signal that enters the decoder in the original transformer is the output signal shifted to the right and masked. This allows the model to auto-regressively complete the signal, similar to recurrent neural networks. Furthermore, the second multi-head attention layer in the decoder is different, because here the keys and queries are taken from the encoder output signal, whereas the values are obtained from the decoder internal signal. After the multi-headed attention layer, a linear layer is used to process the decoder signal into a signal of the desired shape. The final soft-max element is used to convert this signal to a probabilistic scale. In the original use

case, the highest probability value determined which word was selected from the embedding space. For trajectory prediction, this operation is removed because the output can directly be translated into a trajectory via the transformation matrix that was used to normalise the input data. This in fact acts as the embedding space. The number of decoder layers in the original transformer is determined at four, but this hyper-parameter can be changed when iterating on the model architecture. The final model architecture can be found in Appendix B

Because of the desired critical look-ahead time of 3 hours for demand predictions, most flights are not yet airborne. This poses the problem that flights must be predicted without any actual trajectory data points available. It was therefore decided to build the trajectory prediction model primarily for a generative case, relying on flight plan input for both encoder and decoder. Nevertheless, an alternative iteration of the model is made where the transformer is trained for both generative as well as regressive use cases. In this auto-regressive model, the encoder input signal also receives the flight data points that are available at that moment.

Besides the classic transformer neural network, also an adapted version will be created based on empirical testing and evaluation of different transformer neural network layouts. In order to compare the results of the transformer neural network, simple trajectory predictors are built based on a variety of machine learning methods. The test cases include an LSTM network, and a feedforward neural network. Each of these networks will have 4 layers with 2048 neurons. This corresponds to the layer complexity of the classic transformer neural network, hence allowing equal comparison.

D. Model training. After establishing the model layout and parameters, the training phase can begin. Neural networks have variable weights or parameters that each partly contribute to the outcome of the network. With a cost function, the error of the model can be calculated. Via error back-propagation, the specific gradients of each weight can then be determined. The PyTorch library is specifically designed to efficiently apply back-propagation and has several optimisation algorithms available to update the weights accordingly. As a result, there are a lot of variables that can be specified to enhance training, which will be explained in this section.

First of all, a cost function is required that calculates the model error. For the trajectory prediction problem, a custom loss function is specified based on requirements and iterative modelling experience. The cost function is given in Equation 9 and Equation 10. Here λ denotes the root mean square error for a given part of the output data or derivative thereof. The target variable y is set to be the difference between the filed flight plan points and the actual trajectory. It is found that this gives more stable trajectory prediction results than directly predicting the actual trajectory. As a result, the final output vector of the model ($\hat{y}$) must be added to the input signal to get the actual predicted trajectories. W denotes the weights that were assigned to the specific error contribution in the loss function. These weights were determined empirically to emphasise certain parts of the trajectory more during training. Table 2 gives the values of the weights in the cost function. The optimiser used is the Adam optimiser, and the model is trained on an RTX3090 GPU.

$$\lambda = RMSE(\hat{y} - y) \tag{9}$$

$$L = \lambda * W_1 + \lambda' * W_2 + \lambda_{alt} * W_3 + \lambda_{begin} * W_4 + \lambda_{end} * W_5 + \lambda_{cruise} * W_6 \tag{10}$$

Table 2. Loss function weights

Weight	Value	Description
W_1	10	Entire trajectory error
W_2	10	Derivative trajectory error
W_3	10	Altitude error
W_4	3	First three datapoints error
W_5	5	Final 20% of trajectory error
W_6	2	50-70% of trajectory error

E. Demand forecasting. Air traffic sector demand can be defined as the measure of traffic occupation or traffic situation complexity in a specific airspace block during a given time window. Within this research, the simplistic definition of traffic occupancy count is used as demand. To predict the demand of an airspace sector, all crossing flights during the desired time window must be available. For this study, the subject area is the Dutch Flight Information Region (FIR). The traffic passing through this airspace is primarily dependent on flights to and from Schiphol, and other airfields in the Netherlands, but also a portion of crossing traffic is present. Generally, three sources of traffic contribute to the demand: arrivals, departures and crossing flights. Because the FIR reaches up to flight level 245, crossing flights to airports outside the FIR are not a significant contribution, and are therefore left out of scope. Furthermore, based on a preliminary data survey and the research by Könnemann [1], it could be concluded that demand created by departing flights is almost entirely dependent on the take-off time prediction, especially because most departures in this airspace come from airports located within the FIR. As a result, also departures are left out of scope. Hence, only arriving flights in the Dutch FIR are considered for evaluation with the trajectory-based approach.

To predict the FIR demand, it must be known when flights are inside the sector. Algorithm 1, shows the logic applied to obtain the demand from flight objects. Conventionally the demand is calculated via a flight plan defined in 4 dimensions: latitude, longitude, altitude and time. The first phase determines the entry and exit times of traffic in the airspace block. As this research is constrained to arriving flights only, an average sector transit time is taken to determine the exit time. This is calculated to be 25 minutes based on the ADS-B trajectories. This assumption is required because both flight

plan and predicted trajectories do not have the required fidelity to accurately predict the complex Terminal Manoeuvring Area (TMA) vectoring behaviour in this airspace. Average transit times are found to be reasonably accurate and do still allow a fair evaluation of the TP performance impact. The second phase of Algorithm 1 shows how the demand is calculated by counting the entering and exiting flights with a sliding window.

Algorithm 1 Demand prediction

$f_t = [p_1, p_2, ..., p_T]; p_t = [lat, lon, alt, t]$ ▷ 4D flight
$F \leftarrow f$ ▷ Set of flights
S ▷ Airspace
A_t ▷ Arrival times
D_t ▷ Departure times
T_{avg} ▷ Average time in sector
T_{wdw} ▷ Demand window time
T_{start} ▷ Start forecast time
T_{end} ▷ End forecast time
Y_t ▷ Demand forecast
Y_0 ▷ Initial demand at start time

Phase 1 – Determine FIR arrival time

for f_t in F **do**
  for p_t in f_t **do**
    if $p_t[lat, lon, alt]$ in S **then**
      $A_t \leftarrow p_t[t]$
      $D_t \leftarrow p_t[t] + T_{avg}$
      break
    end if
  end for
end for

Phase 2 – Compute demand

$t_0 = T_{start}$
$t_1 = t_0 + T_{wdw}$
while $t_1 <= T_{end}$ **do**
  $A_{wdw} = \sum_0^n A_t[t > t_0 \& t < t_1]$
  $D_{wdw} = \sum_0^n D_t[t > t_0 \& t < t_1]$
  $Y_t \leftarrow Y_0 + A_{wdw} - D_{wdw}$
  $Y_0 = Y_t$
  $t_0 = t_1$
  $t_1 = t_1 + T_{wdw}$
end while

F. Experimental set-up. To evaluate the suitability of the proposed transformer model, both the trajectory prediction as well as the demand forecasting performance must be tested. For this, different experiments are designed in which the transformer models are tested against the conventional flight plan-based approach and the feedforward and LSTM neural networks. The validation dataset consists of 10 full days of traffic that are randomly selected from the month of available data. For the trajectory prediction, this means 2839 flights can be used to assess the TP performance. During training, the predictive accuracy is measured through the loss function (Equation 10). A lower score means a closer correspondence to the target trajectory. However, the loss score is designed specifically to train the model and is not a universal metric to measure trajectory predictive performance. This makes it less suitable for validation. In the experiment, lateral predictive accuracy is measured in along-track, cross-track and horizontal error, as shown in Figure 4. In this research, only the airborne part of the flight is modelled, which means that all trajectories take off at t_0. The errors are then calculated at each timestamp, comparing the actual trajectory to the predicted trajectory. Similarly, the altitude difference at each timestamp is evaluated. The mean absolute error for each metric is taken over the entire trajectory. Another metric that is important to compare is the total trajectory distance, as it shows whether or not the predicted trajectory is globally coherent. In the experiment, the error distribution of the different models will be compared to the baseline trajectory which is the flight plan.

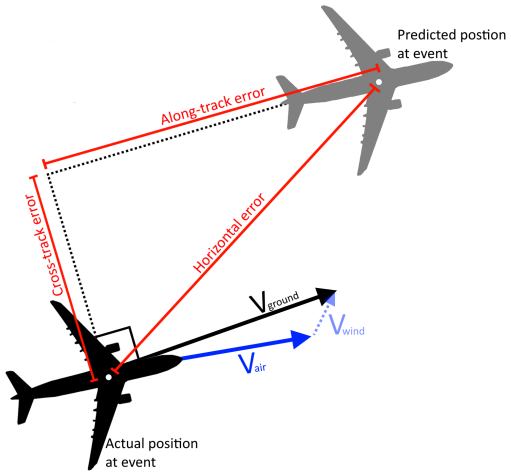


Fig. 4. Horizontal error metrics for trajectory prediction.

The demand prediction experiment considers the predicted trajectories to determine the expected demand in the Amsterdam FIR via Algorithm 1. Arrival time is one of the driving variables, hence this is an important metric to consider. The evaluation of arrival time will be two-folded: Since the predicted trajectories are trained based on aligned take-off times, there is no pre-departure delay predicted by the TP model. This was a considerate design choice because ground-based delay has a significantly different complexity than airborne delay. This was expected to introduce a lot of uncertainty into the model had it been included, while also requiring an architectural change for both the model and data format. Because of this, the trajectory predictor model generates a trajectory that starts at the given departure time in the input dataset. The predictions and actual flights are therefore compared based on flight duration up to the FIR arrival time. This metric is most important to determine the impact of the airborne element on demand forecast. However, when predicting demand, absolute predicted arrival times are the input, hence the absolute arrival time prediction accuracy is evaluated in the experiment as well.

Ultimately, demand forecasts are the most important metric for tactical decision-making. Hence, these are pivotal to evaluate in the experiment as well. In current operations, the demand window is 20 minutes, and it is renewed every 5 minutes. As a result, this experiment considers the same window size and refresh rate. The demand predictions for the 10 days in the test dataset are compared to the actual demand. It is to be noted that this research is scoped to predictions for the situation 3 hours ahead only, and that some flights with limited input or actual trajectory data are filtered out. Hence, the results are not an operational demand forecast, but only an assessment of the 3-hour look-ahead time. However, since this research is constrained to pre-departure generative trajectory predictions, a direct comparative analysis of the demand forecasting performance can still be made.

IV. Results

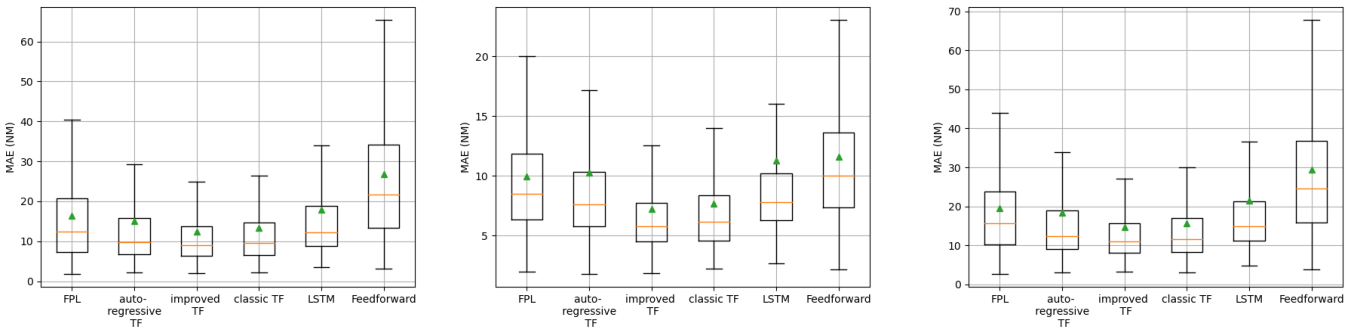
After training several model varieties sufficiently, the model performance can be assessed. This section presents the results of the developed transformer neural network, compared to other methods. In the first subsection, the trajectory prediction performance is assessed. The second subsection shows the performance of the model when used to forecast air traffic demand.

A. Trajectory prediction performance. Before showing the predictive capabilities, a summary of the model training efforts is given in Table 3. With the loss values that were found, it can be expected that the improved transformer seems to be the best-performing model. Moreover, this model also trains relatively fast and is stable up to 2500 epochs.

Table 3. Model training results

	auto-regressive TF	Improved TF	Classic TF	LSTM	Feed-forward
Trainable parameters	24E6	24E6	30E6	117E6	17E6
Time per epoch [s]	33	32	37	39	16
#epochs trained	2500	2500	1600	500	200
Final loss	35	30	39	48	65
Overfitted	No	Yes	No	Yes	Yes

Although training results are important to assess model performance, the experiment on the test dataset will show whether or not the model is capable of improving trajectory predictions. During the first analysis of the experiment, it was found that none of the models produced accurate trajectory predictions for long-haul flights. This limitation was found to be especially relevant for flights longer than 6 hours, which make up approximately 30% of the dataset. therefore it was decided to only include flights that have an estimated flight time below this threshold. The implications of this are further discussed in section V. The final results after having applied this filter are given below. In Figure 5 the mean absolute along-track, cross-track and horizontal error distribution are shown. Figure 6 presents the mean altitude error. Finally, Figure 7 shows the total trajectory distance error.



(a) Along-track error distribution (b) Cross-track error distribution (c) Horizontal error distribution

Fig. 5. Distribution of the lateral trajectory prediction error, where the mean absolute error is taken for every individual flight.

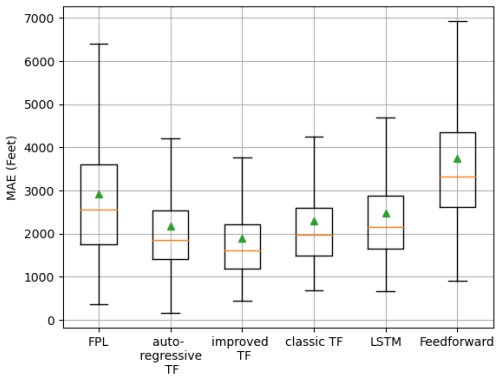


Fig. 6. Mean absolute altitude error of the different TP models.

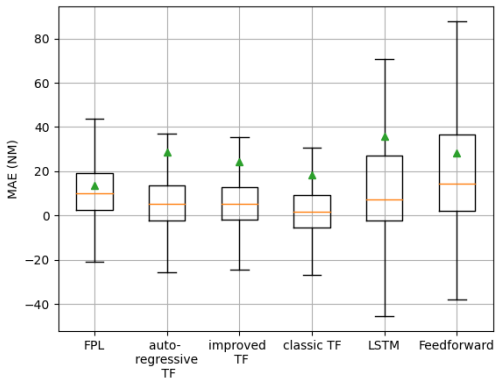


Fig. 7. Mean total trajectory distance error of the different TP models.

When comparing the different models to the predictive performance of the flight plan, it can be noted that the feedforward model has significant difficulty in predicting trajectories. This is to be expected because feedforward neural networks are not usually applied to sequential data structures. The LSTM neural network shows some improvements compared to the flight plan, but during training, it was found to be sensitive to overfitting. After 500 epochs the training was stopped, because validation results became worse. The classic transformer neural network was not sensitive to overfitting but was slower to converge. The final model was trained to 1600 epochs until the loss stabilised, however, no signs of overfitting were found during intermittent validation. The transformer clearly outperforms the three baseline models, with a lower mean error as well as a reduced error distribution. Only for the global distance attribute all machine learning models fail to reduce the mean error, hinting at a small number of trajectories that have very large errors. This is not uncommon for a data-driven model and does not affect the majority of predictions.

Based on literary findings and the obtained results from the three baseline models, the semi-optimised transformer neural network was created through iterative training of various model architectures. In this network, the decoder complexity was reduced to a single layer only, as it was found that the level of complexity in the data was not easily captured through the auto-encoder structure. Yet a linear pipeline performed better and could be trained further. Moreover, the encoder complexity was increased to 6 layers, and the final output block was extended with a linear and relu activation layer. The architecture of the model can be found in Appendix B. The results of this model are a significant improvement from the baseline models. Predictive accuracy both horizontally and vertically improved to a lower median error and better distribution. This shows that the transformer neural network is very capable of explaining the differences between the filed flight plan and actual flight execution. Looking at the altitude predictions, when observing individual trajectories, most of the errors are reduced by more accurately predicting climb and descend phases. The cruise phase prediction is reasonable, but here the flight plan is usually found to still be a bit more accurate. An example of a predicted trajectory is shown in Figure 8. Finally, the improved transformer was adapted to an autoregressive variant that is capable of extrapolating future data points of an already airborne trajectory. Although still providing some improvements in TP accuracy, the model failed to surpass the accuracy of the fully generative models. However, the presented trajectory prediction improvements are overall very significant, especially for the improved generative transformer neural network.

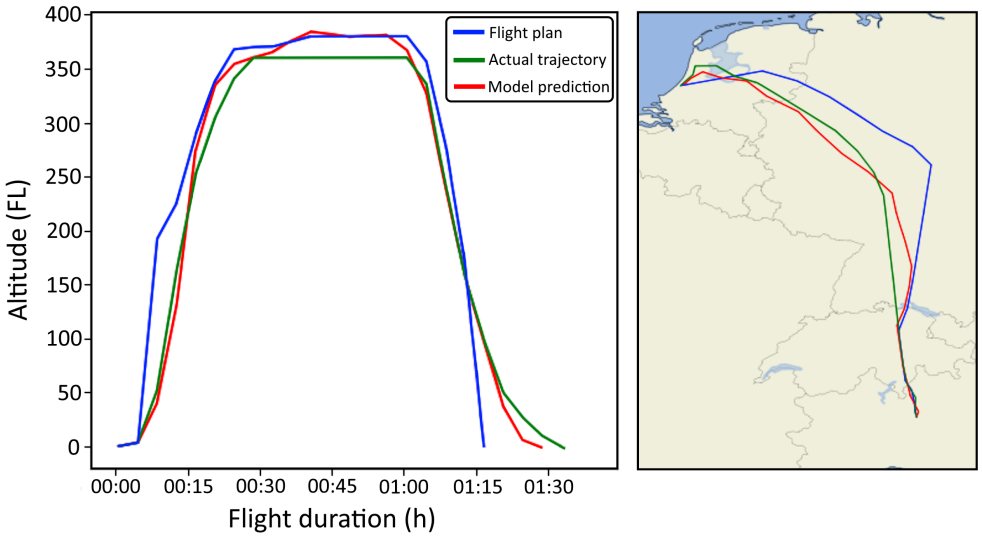


Fig. 8. Example of a prediction for a flight between Milan and Amsterdam. The predicted trajectory is compared to the filed flight plan and the actual trajectory.

B. Demand forecasting performance. With the observed increase in trajectory prediction accuracy, it is to be expected that demand forecasts can benefit from more accurate predictions of aircraft entering the target airspace. Demand forecasting performance is measured with a variety of metrics amongst which the flight duration time up to FIR entry, arrival time and demand. The validation dataset includes 10 days of predicted and actual trajectories that are used to evaluate the performance. First, because only airborne segments of flights are considered, the predicted flight duration times up to FIR entry must be evaluated. This shows whether or not the TP model is likely to make an improved arrival time estimate, and henceforth is suitable for demand forecasting. These results are shown in Figure 9. As expected, all models that showed improved TP accuracy also have a better estimate of the duration of the flight than the flight plan. Secondly, the absolute arrival times at the FIR boundary are calculated with the duration of the flight and the estimated take-off time from the B2B message. These absolute arrival times are then compared to the actual arrival times from the ADS-B trajectories. The absolute arrival time prediction errors are presented in Figure 10.

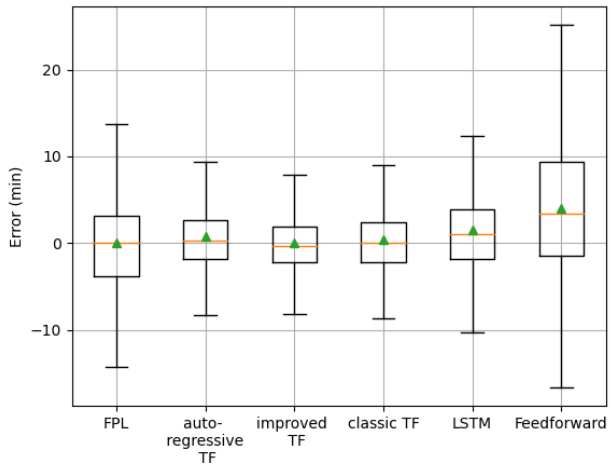


Fig. 9. Error between actual flight duration and the predicted flight duration up until FIR entry.

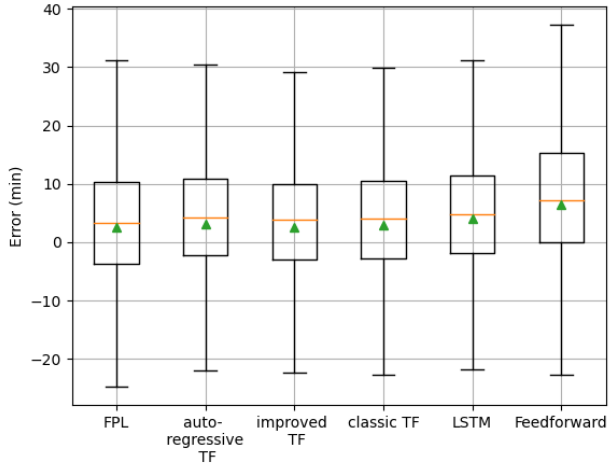


Fig. 10. Error between actual FIR arrival time and the predicted arrival time.

The predicted arrival time of flights by the various models does not show the same increase in accuracy as observed in the flight duration evaluation. This is partly expected because the take-off time uncertainty is known to be a considerable factor, which was argued by Könnemann [1] amongst others. However, some improvements can still be observed, as the spread of errors is reduced to some extent with the transformer models. Finally, the demand forecasts of each method are evaluated. In current operations, LVNL considers the demand forecast at 3 to 5 hours ahead most valuable. This allows the ANSP to apply the required capacity resources or to regulate incoming traffic. For this reason, the dataset is constrained to B2B messages at 3 hours before the actual arrival of the flight. The demand forecast results of all different trajectory predictor models are presented in Table 4.

Table 4. Demand forecast model results. Measured by the difference in the number of flights that were predicted to enter the airspace and the actual number of flights.

	RMSE [flights]	MAE [flights]	R2	Std. Deviation [flights]
FPL	1.83	1.12	0.79	1.83
Autoregressive TF	1.73	1.07	0.81	1.73
Improved TF	1.66	1.04	0.82	1.66
Classic TF	1.78	1.10	0.80	1.78
LSTM	1.83	1.13	0.78	1.83
Feedforward	2.1	1.32	0.72	2.09

Comparing the demand errors from the flight plan-based method to the tested TP models, it becomes clear that most models do not provide any significant improvements. The feedforward model performs poorly, which was expected based on the trajectory prediction accuracy. The LSTM model did have slightly better TP accuracy compared to the flight plan, but these improvements do not lead to any benefits in the demand forecasting case. The transformer neural networks however do show slightly improved demand forecasts. Particularly, the improved transformer shows a more significant jump over the flight plan-based method. The improved transformer model has a root mean square error that is 10% lower than the flight plan-based approach. Compared to the results of the other models, this is the only significant improvement, given the dataset size of 10 days. When looking at the demand error distribution in Figure 11, it becomes clear that the improved transformer marginally reduces the peak errors. Hence, the improved trajectories potentially provide a more stable demand forecast. Nevertheless, the differences are very small. To further clarify the results, Figure 12 shows the demand forecast for an entire day of traffic. The predictions of the traffic peaks seem to be less erroneous compared to the flight plan-based approach, which is a desirable improvement. The remaining demand forecast figures are presented in Appendix D

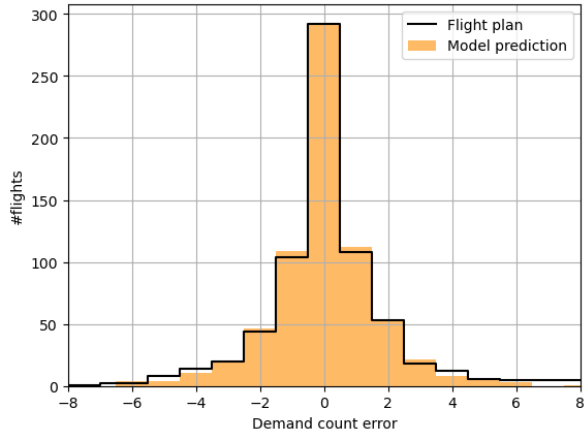


Fig. 11. Distribution of predicted demand error for the flight plan and the improved transformer approach.

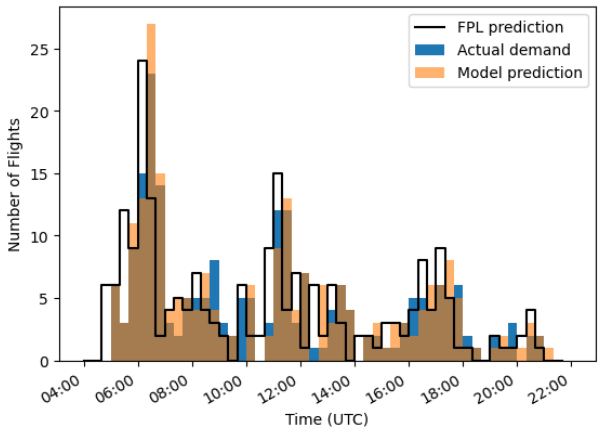


Fig. 12. Example of predicted demand on May 19 2021, where each bin is predicted with a 3-hour look-ahead time.

V. Discussion

In this discussion, the results of the proposed machine learning models are evaluated in both trajectory prediction and demand forecasting. First, the suitability of the results concerning the trajectory and demand prediction domain are discussed. Secondly, the implications and limitations of the chosen methodology are considered. Finally, recommendations for continuing these research efforts will be presented.

A. Suitability of generative machine learning methodology to trajectory prediction. When examining the results of the trajectory prediction models, it was observed that the generative approach failed to predict accurately when flight time exceeded 6 hours. This can be explained by the pre-processing steps taken, because all flights are normalised between 0 and 1. This means that the spatial resolution is reduced for flights with more data points. A different normalisation approach was tested, which normalised trajectories over the total trajectory distance, but this did not lead to a model that converged during training. Nonetheless, the performance of the predictor for flights below the 6-hour threshold was found to be very good. As explained in the results, the feedforward neural network failed to accurately predict trajectories, which is not unexpected given the sequential nature of the input data. The LSTM network performed well, but the transformer and improved transformer both delivered even better results, proving that a generative data-driven approach can bring a lot of value to long-term trajectory prediction problems. The auto-regressive model was expected to provide even better results, but this did not prove to be the case. This can be partly explained by the fact that most flights within the 6-hour duration filter did not have many airborne data points, meaning the model does not receive much contextual data. However, it could also be that the model architecture was chosen wrongly. During the design phase, it was found that the transformer is difficult to train and can converge much better or worse with a different layout. It is therefore believed that the generative model can be optimised further in the next iteration. When placing the results into an academic context, Liu & Hansen [10] also created a generative LSTM neural network to predict trajectories on a single city pair with a flight duration of around 3 hours. Although differences exist, a comparison in TP accuracy can still be drawn. Liu & Hansen obtained a mean absolute horizontal error of 50 nautical miles and a vertical error of 2860ft. The proposed transformer model outperforms this approach on a large set of diverse flights. With the improved transformer reaching a mean absolute horizontal error of 16 nautical miles, and a vertical error of 2000ft.

Nonetheless, the trajectory cumulative distance error for all methods was found to increase compared to the flight plan. After examining a random draw of flights, it was found that predicted trajectories did still see occasional jumps or whimsy behaviour. Examples of this can be found in Appendix C. As a mitigating measure, the training function was adapted to include the trajectory derivative, meaning vertical and lateral velocity vectors. Yet it did not constrain all predictions sufficiently. Furthermore, the highly dynamic flight behaviour in the departure and arrival phases also causes whimsy characteristics of the predictions. These patterns are introduced in the data by operational procedures and vectoring by air traffic control. However, these effects mainly apply to the first and final stages of the trajectories. For demand applications, the effect on arrival is less relevant, as the trajectory is only evaluated up to the FIR entry point.

In conclusion, machine learning methods that are designed to process sequential data show improved predictive performance when generating new trajectories based on the available B2B flight data. Especially the transformer neural network showed a significant step up. Although training took longer than the LSTM network, the training was much more stable and provided better final results. It is difficult to pinpoint the most important element that makes the method work. However, a few observations were made during the modelling phase. First and foremost the data formatting contributed significantly to the success of the transformer and LSTM models. The data variance and dimensionality were reduced with trajectory normalisation, leading to better training. Moreover, the decision to train on the difference between the actual and planned flight path was also very effective. In the modelling phase, different data formats were tested, after which this method gave the best training results. The same holds for training methodology, where a custom loss function provided a big increase in training effectiveness. The data format and loss function can be applied to any machine learning method, however, the improved transformer model did outperform the other methods. Hence, model-specific elements contribute as well. In part, the multi-headed attention mechanism can be regarded as the driving element of the transformer, being much less prone to vanishing gradient problems. This results in more stable and longer training. Finally, omitting the complex encoder in the improved transformer was also a successful contribution. Training an auto-encoder through back-propagation is difficult for large data structures. The encoder may become a bottleneck as the dimension of the input data is lowered. The linear pipeline therefore proved more effective, but in a future iteration, the encoder output size can also be increased to still implement the encoder-decoder structure successfully.

B. Suitability of predicted trajectories to air traffic demand prediction. Given the improved trajectory predictions, it was possible to assess the impact on demand forecasts for the 3-hour look-ahead time. Although the test dataset was not as large as desired, still some valuable results could be found. When evaluating the airborne segment only, the flight duration time estimate did indeed become more accurate when trajectories were predicted by the successful TP models. The interquartile range of the improved transformer was reduced by almost 30% with respect to the flight plan-based approach. However, when comparing actual arrival times to the predicted arrival times, the improvements seemed to wash out compared to the flight plan-based ap-

proach. This was unexpected, but a potential explanation can be that the model performs worse on more critical flights. In demand prediction and optimisation models such as those proposed by Fernández et al.[3] & Sridhar et al.[20], departure time is a driving variable for the sector demand. However, departure time is not changed by the model, but it is commonly argued that ground-based delay will result in different flight behaviour. Tielrooij et al.[21] state that pilots tend to make up for any previous delay by being more assertive in asking for direct routes and increasing the cruise speed. Although the model received up-to-date departure time and delay information as input data, it can still be that these patterns are not fully captured by the neural network. Furthermore, if regulations were imposed on flights within three hours before arrival, this could also change the actual arrival time without the model knowing about the induced delay. All these uncertainties may lead to less accurate predictions for flights with larger deviations than flights that adhere more strictly to the flight plan. This results in better arrival time predictions for standard flights, but worse predictions for non-standard flight behaviour. It is uncertain what the root cause of this problem is; either the chosen neural networks are not capable of explaining possible correlations between flight path changes and ground delay, or the training dataset was too small to provide enough samples of flights that suffer this problem. Due to time and computational limitations, it was not possible to further investigate these hypotheses.

Moreover, the actual demand forecasts of the different trajectory predictors have been assessed. This showed that indeed most models did not provide a statistical increase in demand forecast accuracy. However, the improved transformer neural network did show a more significant increase in demand prediction accuracy. This means that a small improvement in arrival time error may lead to more stable demand forecasts. Comparing the distribution of the error, it can be seen that the new predictions produce slightly lower peak errors, meaning real odd predictions are reduced. Examining the demand forecast during the day, it was found that peak demand predictions especially were improved over the flight plan method. For the air traffic controller supervisor, the peaks are more important to be predicted right, as this can cause a loss of capacity. Better peak predictions can immediately lead to more effective and efficient flow management. The results are therefore promising, but to confirm the improved peak predictions, the test dataset must be increased considerably.

C. Implications and limitations. With the given results, it can be concluded that the chosen generative trajectory prediction approach offers considerable improvements, especially with the semi-optimised transformer neural network. The primary limitation of the method is that the trajectory prediction accuracy reduces drastically for flights with a duration of more than 6 hours. This can be circumnavigated by training a separate model for long-haul flights. This will likely capture the lower spatial resolution introduced with the normalisation algorithm.

When using the predictions for demand forecasting applications, seemingly improved and more stable results are observed in the test dataset. However, as discussed before, without reliable departure time availability, the demand forecasts cannot drastically be improved. LVNL [22] has built a random forest regressor that predicts the time offset for arriving aircraft. This method is trained on flight schedule information, which obtains a mean absolute error reduction of 2 minutes at the 3-hour look-ahead time. In this methodology, the ground process delay is not excluded, which possibly explains the significant error reduction. Unfortunately, similar results were not obtained by improving the TP. Nonetheless, the overall error distribution of demand forecasts was decreased. Furthermore, for airborne flights, the actual take-off time was explicitly left out of the analysis. This would directly increase the demand prediction performance, but it blocks a pure assessment of the trajectory predictor influence.

Finally, although no very significant improvements in arrival time predictions were found compared to the more generalised random forest regressor approach, it is still worth mentioning that the trajectory-based approach has an interesting additional capability. In the end, the supervisor applies the demand forecast to estimate the future task load of the controller. Trajectory information can be of high value when estimating the task load of a controller. One can imagine that complex converging flight paths require much more cognitive skills of a controller than parallel or well-separated trajectories. Therefore, the predicted trajectories can in a future process iteration lead to a task load model directly. The observed trajectory prediction improvements therefore imply that the proposed generative transformer neural network is very suitable for both TP and demand forecasting tasks, which will lead to more effective tactical interventions in the ATM system.

D. Recommendations. The proposed transformer neural network methodology showed a considerable increase in the accuracy of generative trajectory prediction. This led to a slight improvement in demand forecasting, but this could not be validated statistically. This is mainly a result of the dataset that was constrained to one month of traffic. The first recommendation is therefore to increase the dataset, preferably to a year of traffic. This will not only allow for a larger test sample to confirm the hypothetical gains, but in addition allows training the transformer TP model even further. Although the influence of COVID in the dataset was assumed to be negligible for trajectory prediction, a new dataset can also exclude that potential influence. Nonetheless, it must be noted that already with a month of data, the computational resources were a limiting factor, despite the availability of a high-end system.

Secondly, to improve demand forecasts, the departure time uncertainty is undeniably the largest source of error. The

trajectory-based approach can be applied in parallel with an improved departure time estimate. Corrective models such as the random forest model as applied by LVNL [22] may be investigated to complement the predicted trajectories. A combined approach could leverage the benefits of using more accurate trajectories for advanced demand forecasting that includes traffic complexity metrics. Also, in the future TBO concept, trajectories may be managed and negotiated between the airspace user and ANSP [21]. A combined model that predicts departure times and generates trajectories is a very interesting path of development in that regard. Despite that, it is recommended to also investigate aggregate demand prediction models such as the graph neural network that was presented by Ma et al.[7]. This approach does show a statistical decrease in flow prediction error, which can result in better demand and capacity balancing without a fully developed TBO concept. Different machine learning models can be investigated to predict air traffic demand. The graph network is a good option to model network-wide flow and demand interaction between airspace sectors. For single-sector predictions, the transformer neural network or a variety thereof can also be used to merely predict traffic flow or demand.

Although the transformer was found to be very capable of generating more accurate trajectories, it is recommended to continue investigating different varieties or types of machine learning models to improve the TP performance. Token mixing models, such as the Fourier transformation-based neural network that was proposed by Lee-Thorp et al.[17], can be a promising alternative to the transformer neural network. Specifically, the training and processing speed may be increased with more efficient machine learning methods.

VI. Conclusion

In order to pursue more sustainable, safe and efficient airspace navigation, the global ATM system is developing towards the concept of Trajectory Based Operations. ANSPs such as LVNL seek to manage flight trajectories in all aspects of their business. One of these aspects is to make sure air traffic demand can meet the available capacity in the airspace. This is not a trivial task, as flight plans and schedules do not always give the most reliable demand picture when planning 3 to 5 hours ahead of time. The goal of this research was therefore to improve air traffic sector demand forecasting, by exploring machine learning-based trajectory prediction.

To start, in section II, an extensive literature review was performed on the three primary aspects. Demand forecasts can be made with a more general aggregate approach that considers flight schedules and flows. Alternatively, 4-dimensional flight paths can be used to estimate the demand. The latter approach fits the TBO concept best. Henceforth, trajectory prediction was examined, where a generative data-driven approach proved to be the most promising methodology. Finally, machine learning methods were reviewed after which the transformer neural network was selected as a novel and promising method.

In section III the methodology was presented. For the input data, flight information messages from the Eurocontrol NM B2B feed at LVNL were gathered. This dataset includes the filed flight plan and flight status information at the desired look-ahead time of 3 hours before arrival. Furthermore, the OpenSky ADS-B repository was consulted to retrieve actual flight trajectories. Data was prepared and compared based on duration (aligned take-off times). This allowed a direct comparison of trajectory prediction performance, without including ground delay interference. In particular, the applied normalisation techniques contributed to the generative performance of the model.

Consecutively, the trajectory prediction and demand forecasting models were built. The evaluated TP models that were selected were a conventional transformer neural network [13], but also an LSTM and feedforward neural network of similar layer complexity. Based on these findings, an improved version of the transformer was built, as well as an auto-regressive version. For the demand forecasting evaluation, a flow aggregate model was created that checked flights for entering and leaving the Dutch FIR airspace. The obtained inflow and outflow were summed with the demand in the previous window.

In section IV, the results of the trajectory prediction models were compared and analysed. Although the feedforward network failed to converge, the other models showed improvements in the predictive accuracy of the flight plan. Especially the transformer neural networks showed very stable training behaviour and could be optimised to make significant improvements. Unfortunately, this only holds for flights with medium duration up to 6 hours of flight time. The chosen data normalisation method is likely the problem here, so improvements can be expected with a dedicated model.

After analysing demand forecasts and flight arrival times in section V, it can be concluded that the improved trajectories did result in better flight time estimates. Yet, for actual arrival time prediction, these improvements washed out slightly. Only the semi-optimised transformer network showed a more significant increase in demand forecasting performance with reduced error outliers and better peak traffic prediction. The results, however, can be improved directly by using available take-off time when a flight is airborne, but this holds for all trajectory-based methods. For demand prediction, the uncertainty in departure times is a driving parameter, but efforts made to improve this estimate will likely also complement the trajectory-based approach that is presented in this research.

References

1. Erik Könnemann. Performance impact of improved departure time prediction relative to sector demand & arrival time predictability. *TU Delft Repository*, December 2014.
2. Eugene Gilbo and Scott Smith. A new model to improve aggregate air traffic demand predictions. *AIAA Guidance, Navigation and Control Conference and Exhibit*, August 2007. doi: 10.2514/6.2007-6450.
3. Esther Calvo Fernández, José Manuel Cordero, George Vouros, Nikos Pelekis, Theocharis Kravaris, Harris Georgiou, Georg Fuchs, Natalya Andrienko, Gennady Andrienko, Enrique Casado, et al. Dart: A machine-learning approach to trajectory prediction and demand-capacity balancing. *Seventh SESAR Innovation Days, Belgrade*, November 2017.
4. Daniel Delahaye, Adrían García, Julien Lavandier, Supatcha Chaimatanan, and Manuel Soler. Air traffic complexity map based on linear dynamical systems. *Aerospace*, 9(5), April 2022. ISSN 2226-4310. doi: 10.3390/aerospace9050230.
5. Banavar Sridhar, Gano Chatterji, Kapil Sheth, and Tarun Soni. An aggregate flow model for air traffic management. *Journal of Guidance Control and Dynamics*, 29:992–997, July 2006. doi: 10.2514/1.10989.
6. Padmanabhan K. Menon, Gregory D. Sweriduk, and Karl Bilimoria. New approach for modeling, analysis, and control of air traffic flow. *Journal of Guidance Control and Dynamics*, 27:737–744, August 2002.
7. Chunyao Ma, Sameer Alam, Qing Cai, and Daniel Delahaye. Sector entry flow prediction based on graph convolutional networks. In *International Conference on Research in Air Transportation*, 2022.
8. Junzi Sun. *Open Aircraft Performance Modeling: Based on an Analysis of Aircraft Surveillance Data*. PhD thesis, Delft University of Technology, June 2019.
9. Jean-Luc Overkamp. Long short-term memory network based trajectory prediction incorporating air traffic dynamics. *TU Delft Repository*, September 2021.
10. Yulin Liu and Mark Hansen. Predicting aircraft trajectories: A deep generative convolutional recurrent neural networks approach. December 2018. doi: 10.48550/ARXIV.1812.11670.
11. Bart Rozendaal. A neural network approach in optimising airport strategy with trajectory prediction. Technical report, Knowledge and Development Center Mainport Schiphol, July 2022.
12. You Wu, Hongyi Yu, Jianping Du, Bo Liu, and Wanting Yu. An aircraft trajectory prediction method based on trajectory clustering and a spatiotemporal feature network. *Electronics*, 11:3453, October 2022. doi: 10.3390/electronics11213453.
13. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., December 2017.
14. Shuang Zhang, Rui Fan, Yuti Liu, Shuang Chen, Qiao Liu, and Wanwen Zeng. Applications of transformer-based language models in bioinformatics: a survey. *Bioinformatics Advances*, 3(1), January 2023. ISSN 2635-0041. doi: 10.1093/bioadv/vbad001. vbad001.
15. Zhibo Wang, Jiayu Guo, Zhengming Hu, Haiqiang Zhang, Junping Zhang, and Jian Pu. Lane transformer: A high-efficiency trajectory prediction model. *IEEE Open Journal of Intelligent Transportation Systems*, 4:2–13, January 2023. doi: 10.1109/OJITS.2023.3233952.
16. Lina Achaji, Thierno Barry, Thibault Fouqueray, Julien Moreau, Francois Aioun, and Francois Charpillet. Pretr: Spatio-temporal non-autoregressive trajectory prediction transformer, March 2022.
17. James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontañón. Fnet: Mixing tokens with fourier transforms. *CoRR*, abs/2105.03824, May 2022. doi: 10.48550/arXiv.2105.03824.
18. Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision, May 2021.
19. Phu N. Tran, Hoang Q. V. Nguyen, Duc-Thinh Pham, and Sameer Alam. Aircraft trajectory prediction with enriched intent using encoder-decoder architecture. *IEEE Access*, 10: 17881–17896, February 2022. doi: 10.1109/ACCESS.2022.3149231.
20. Banavar Sridhar, Neil Chen, and Hok Ng. An aggregate sector flow model for air traffic demand forecasting. *9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, September 2009. doi: 10.2514/6.2009-7129.
21. Maarten Tielrooij, Robert Kok, Theo de Jong, Ferdinand Dijkstra, Tim Dufourmont, Esther Lap, Ander Okina, and Reinier Vos. Transition to trajectory based operations (tbo). Technical report, Knowledge & Development Centre Mainport Schiphol, February 2022.
22. Luchtverkeersleiding Nederland. *Software Design Description (SDD), Decision Support Tool - Load Predictor*, 2.0 edition, 2020.
23. Hans Koolen and Ioana Coliban. Flight progress messages document. *"Eurocontrol Network Manager"*, 2.600:p.1–157, July 2020.

A. Data

In Figure 13, an excerpt of the B2B data that was used to construct the input dataset is shown. This data was kindly provided by LVNL, and was pre-processed from the original data feed from Eurocontrol. Not all data fields from the flight progress messages have therefore been parsed into this dataset. Furthermore, not all fields are consistently filled with data. For example, some timing information is not available in this specific dataset. This can be seen in the empty columns. However, the architecture is present, so once this data does become available, these fields can directly be included. The first row of data clarifies the contents of the column. For a thorough review of all variables, one can consult the flight progress message manual from Eurocontrol [23].

id	flight_plan_id	ifpl_id	insertion_timestamp	message_timestamp	flfpl_id	acid	adep	ades	ata	cta
Message identification number	2021********_****_ADEP_ADES	AA******76	YYY-MM-DD HH:MM:SS	YYY-MM-DD HH:MM:SS	2021********	KLM***	ADEP	EHAM	YYY-MM-DD HH:MM:SS	YYY-MM-DD HH:MM:SS
5151aa817-e5cpp-3c4b-15zf2-84zk	2021********_****_K***_EHAM	AA******73	2021-05-DD HH:MM:SS	2021-05-DD HH:MM:SS	2021********	DAL***	K***	EHAM	2021-05-DD HH:MM:SS	
hj2ad982-p337-2f33-6bmw-10235c	2021********_****_W***_EHAM	AA******38	2021-05-DD HH:MM:SS	2021-05-DD HH:MM:SS	2021********	GIA***	W***	EHAM	2021-05-DD HH:MM:SS	
cfd35abc-1296-6j21-a320-4995oja4	2021********_****_E***_EHAM	AA******32	2021-05-DD HH:MM:SS	2021-05-DD HH:MM:SS	2021********	EJU****	E***	EHAM	2021-05-DD HH:MM:SS	
882a7b53-495-8yp32-b17z-az67a3	2021********_****_L***_EHAM	AA******67	2021-05-DD HH:MM:SS	2021-05-DD HH:MM:SS	2021********	HV***	L***	EHAM	2021-05-DD HH:MM:SS	

eta	wtc	aobt	cobt	eobt	message	taxiTime	airspaces	atfmDelay	modelType	flightType	readyState
YYY-MM-DD HH:MM:SS	H	YYY-MM-DD HH:MM:SS	YYY-MM-DD HH:MM:SS	YYY-MM-DD HH:MM:SS	message ID & timestamp	HH:MM:SS	eti [entry times] xti [exit times] airspaceId	HH:MM:SS	Type of flight model	Type of flight	Flight readiness state
2021-05-DD HH:MM:SS	M			2021-05-DD HH:MM:SS	{id,timestamp}	00:10:00	[eti's, xti's, airspaceId's]	00:00:00	EST	U	IN
2021-05-DD HH:MM:SS	H			2021-05-DD HH:MM:SS	{id,timestamp}	00:10:00	[eti's, xti's, airspaceId's]	00:00:00	ACT	U	IN
2021-05-DD HH:MM:SS	M			2021-05-DD HH:MM:SS	{id,timestamp}	00:10:00	[eti's, xti's, airspaceId's]	00:00:00	EST	U	IN
2021-05-DD HH:MM:SS	M			2021-05-DD HH:MM:SS	{id,timestamp}	00:20:00	[eti's, xti's, airspaceId's]	00:00:00	CAL	U	IN

flightRules	flightState	regulations	routePoints	aircraftType	discrepancies	airlineOperator	aircraftRegistration
flight rules	ATFM status	List of regulations applied to the flight	**Filed flight plan** eto: [estimated time over] lat: [latitude] lon: [longitude] type: [type of waypoint] pointId: [name of waypoint] pointRoute: [routename if wpt is on route] aerodromeId: [if wpt is AD] flightLevel: [planned flightlevel]	B78X	{'offblock': False, 'aircraftType': False, 'aircraftRegistration': False}	Responsible airline operator	Airframe registration
IFR	TA	[]	[eto,lat,lon,type,pointId, etc.]	A339	{'offblock': False, 'aircraftType': False, 'aircraftRegistration': False}	DAL	N****
IFR	AA	[]	[eto,lat,lon,type,pointId, etc.]	B77W	{'offblock': False, 'aircraftType': False, 'aircraftRegistration': False}	GIA	PK***
IFR	AA	[]	[eto,lat,lon,type,pointId, etc.]	A32N	{'offblock': False, 'aircraftType': False, 'aircraftRegistration': False}	EJU	OE***
IFR	AA	[]	[eto,lat,lon,type,pointId, etc.]	B738	{'offblock': False, 'aircraftType': False, 'aircraftRegistration': False}	HV	PH***

mostPenalizingRegulation	operatingAircraftOperator	atot	tobt	tsat	ttot	status	departureAirportType
Regulation name that is most constraining	Actual operating airline	YYY-MM-DD HH:MM:SS	YYY-MM-DD HH:MM:SS	YYY-MM-DD HH:MM:SS	YYY-MM-DD HH:MM:SS	ACDM status	CDM or STANDARD departure airport
						DEPARTING_FROM_STANDARD_AIRPORT	STANDARD
	GIA					DEPARTING_FROM_STANDARD_AIRPORT	STANDARD
						DEPARTING_FROM_STANDARD_AIRPORT	STANDARD
	HV					DEPARTING_FROM_STANDARD_AIRPORT	STANDARD

Fig. 13. The above table presents an anonymised excerpt of the Eurocontrol Network Manager B2B dataset that was used to construct the input dataset.

B. model architecture

This appendix shows the PyTorch modules of the various models used to evaluate the predictive performance of the chosen machine learning methods.

```
TimeSeriesTransformer_classic(
  (encoder_input_layer): Linear(in_features=19, out_features=512, bias=True)
  (positional_encoding_layer): PositionalEncoder(
    (dropout): Dropout(p=0, inplace=False)
  )
  (encoder): TransformerEncoder(
    (layers): ModuleList(
      (0-3): 4 x TransformerEncoderLayer(
        (self_attn): MultiheadAttention(
          (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
        )
        (linear1): Linear(in_features=512, out_features=2048, bias=True)
        (dropout): Dropout(p=0, inplace=False)
        (linear2): Linear(in_features=2048, out_features=512, bias=True)
        (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        (dropout1): Dropout(p=0, inplace=False)
        (dropout2): Dropout(p=0, inplace=False)
      )
    )
  )
  (encoder_linear): Linear(in_features=512, out_features=512, bias=True)
  (decoder_norm_layer): LayerNorm((4,), eps=1e-05, elementwise_affine=True)
  (decoder_input_layer): Linear(in_features=4, out_features=512, bias=True)
  (decoder): TransformerDecoder(
    (layers): ModuleList(
      (0-3): 4 x TransformerDecoderLayer(
        (self_attn): MultiheadAttention(
          (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
        )
        (multihead_attn): MultiheadAttention(
          (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
        )
        (linear1): Linear(in_features=512, out_features=2048, bias=True)
        (dropout): Dropout(p=0, inplace=False)
        (linear2): Linear(in_features=2048, out_features=512, bias=True)
        (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        (norm3): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        (dropout1): Dropout(p=0, inplace=False)
        (dropout2): Dropout(p=0, inplace=False)
        (dropout3): Dropout(p=0, inplace=False)
      )
    )
  )
  (linear_mapping): Linear(in_features=512, out_features=4, bias=True)
)
```

Fig. 14. PyTorch architecture of the classic transformer neural network.

```
LSTM_Model(
  (linear_pre): Linear(in_features=19, out_features=4, bias=True)
  (lstm): LSTM(4, 2048, num_layers=4, batch_first=True)
  (linear_post3): Linear(in_features=2048, out_features=4, bias=True)
)
```

Fig. 15. PyTorch architecture of the LSTM neural network.

```
FeedForward  Model(
  (linear_pre): Linear(in_features=19, out_features=2048, bias=True)
  (hidden): ModuleList(
    (0): Linear(in_features=2048, out_features=2048, bias=True)
    (1): Sigmoid()
    (2): Linear(in_features=2048, out_features=2048, bias=True)
    (3): Sigmoid()
    (4): Linear(in_features=2048, out_features=2048, bias=True)
    (5): Sigmoid()
    (6): Linear(in_features=2048, out_features=2048, bias=True)
    (7): Sigmoid()
  )
  (linear_post): Linear(in_features=2048, out_features=4, bias=True)
)
```

Fig. 16. PyTorch architecture of the feedforward neural network.

```
TimeSeriesTransformer_v2(
  (encoder_input_layer): Linear(in_features=19, out_features=512, bias=True)
  (positional_encoding_layer): PositionalEncoder(
    (dropout): Dropout(p=0, inplace=False)
  )
  (encoder): TransformerEncoder(
    (layers): ModuleList(
      (0-5): 6 x TransformerEncoderLayer(
        (self_attn): MultiheadAttention(
          (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
        )
        (linear1): Linear(in_features=512, out_features=2048, bias=True)
        (dropout): Dropout(p=0, inplace=False)
        (linear2): Linear(in_features=2048, out_features=512, bias=True)
        (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        (dropout1): Dropout(p=0, inplace=False)
        (dropout2): Dropout(p=0, inplace=False)
      )
    )
  )
  (encoder_linear): Linear(in_features=512, out_features=512, bias=True)
  (decoder_norm_layer): LayerNorm((4,), eps=1e-05, elementwise_affine=True)
  (decoder_input_layer): Linear(in_features=4, out_features=512, bias=True)
  (decoder): TransformerDecoder(
    (layers): ModuleList(
      (0): TransformerDecoderLayer(
        (self_attn): MultiheadAttention(
          (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
        )
        (multihead_attn): MultiheadAttention(
          (out_proj): NonDynamicallyQuantizableLinear(in_features=512, out_features=512, bias=True)
        )
        (linear1): Linear(in_features=512, out_features=2048, bias=True)
        (dropout): Dropout(p=0, inplace=False)
        (linear2): Linear(in_features=2048, out_features=512, bias=True)
        (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        (norm3): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        (dropout1): Dropout(p=0, inplace=False)
        (dropout2): Dropout(p=0, inplace=False)
        (dropout3): Dropout(p=0, inplace=False)
      )
    )
  )
  (linear_mapping1): Linear(in_features=512, out_features=512, bias=True)
  (relu): ReLU()
  (linear_mapping2): Linear(in_features=512, out_features=4, bias=True)
)
```

Fig. 17. PyTorch architecture of the improved transformer neural network used for generative trajectory prediction, as well as for the auto-regressive model.

C. Trajectories

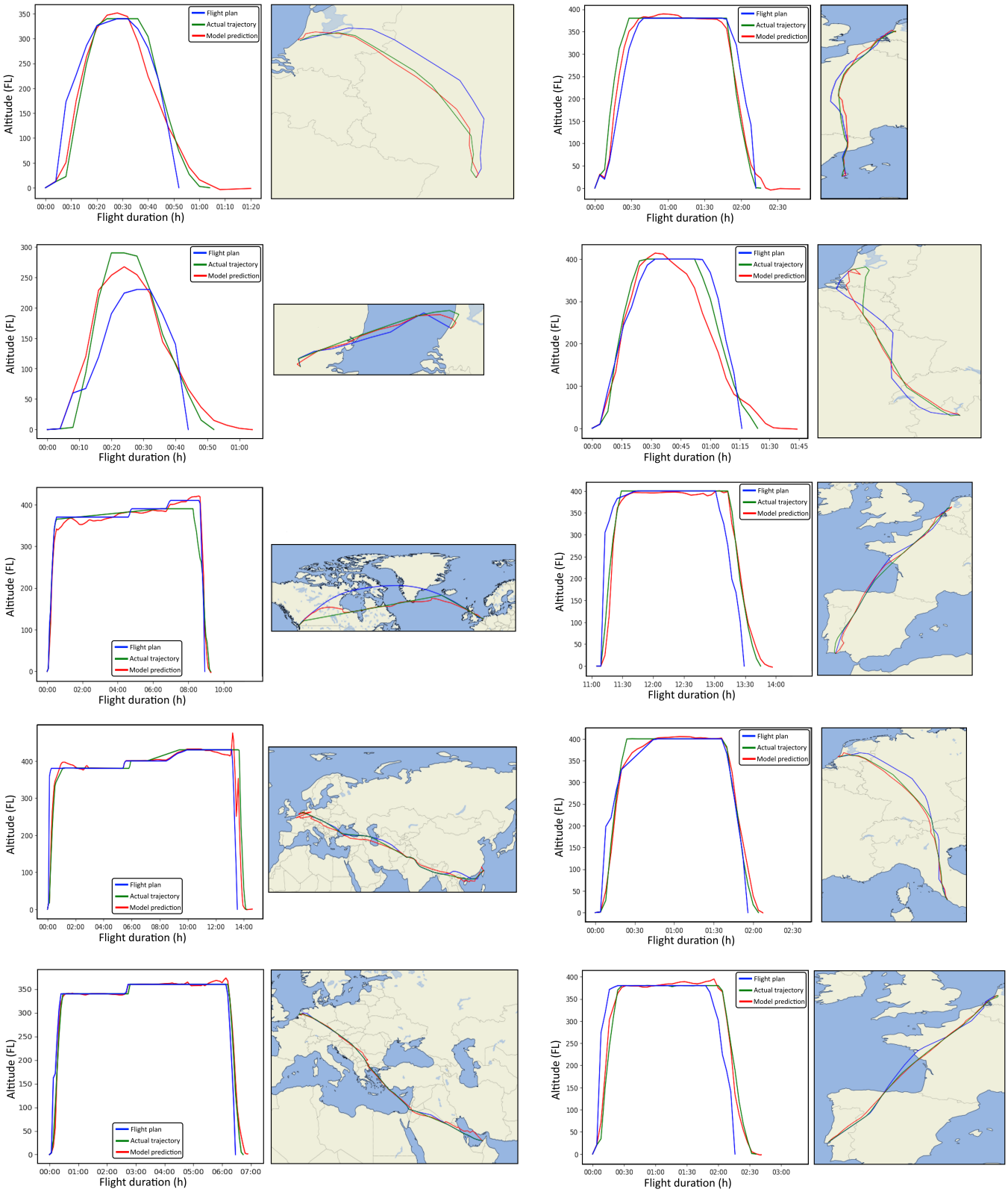


Fig. 18. Various predicted trajectories from the validation dataset.

D. Demand forecasts

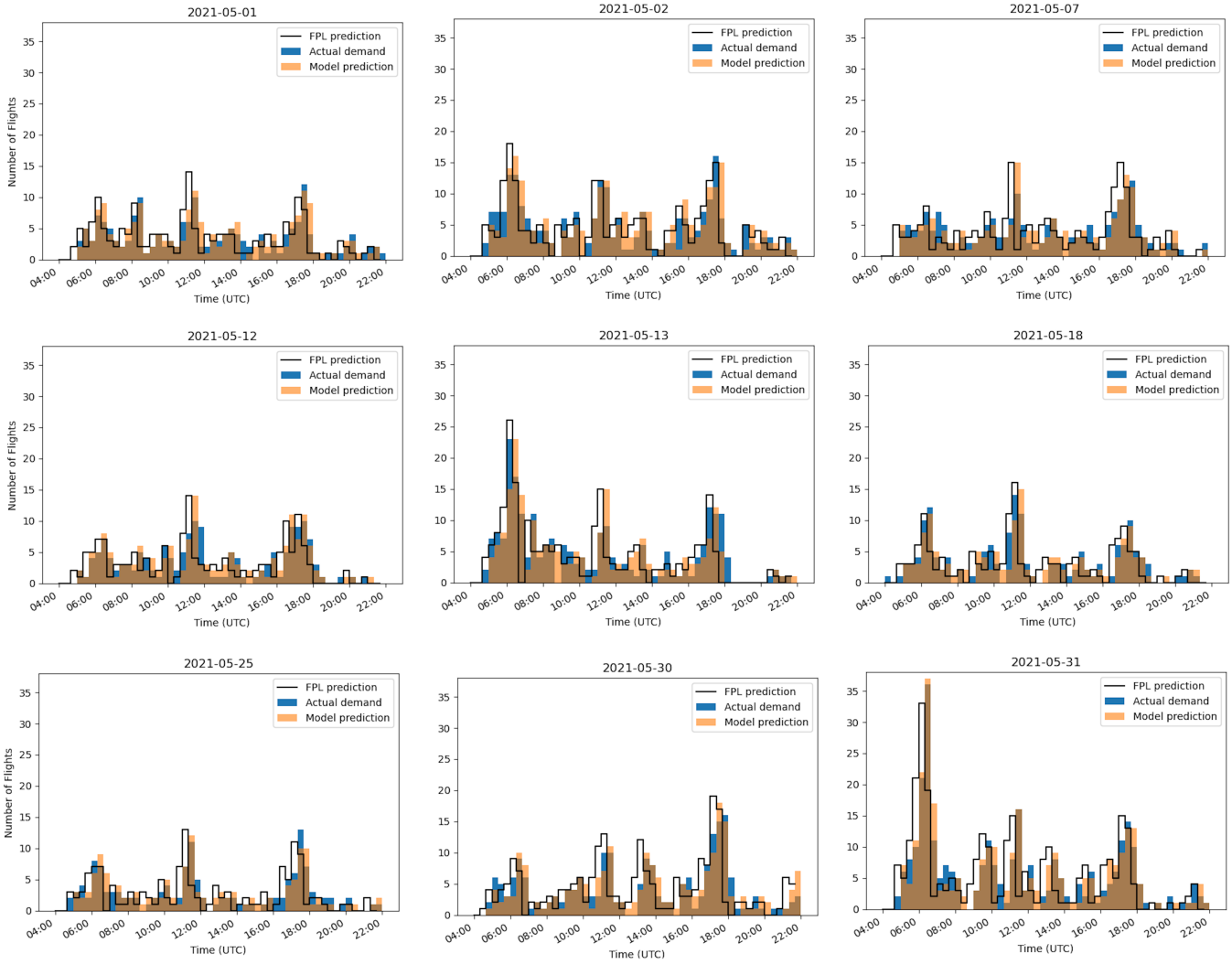


Fig. 19. Demand forecast comparison of all dates in the test dataset. Note that the predictions for May 19 2021 are given in Figure 12

Part II

Preliminary Analysis

1 Introduction

Over the last century, aviation has grown from a distant dream of the pioneers, longing to soar the skies, to a reality where thousands of aircraft bring people to every corner of the earth. Although air transport has brought humanity many good things, there are huge challenges ahead to keep the dream of flying available for the generations to come. The air transport sector must become more sustainable and reduce its footprint on the planet. This drive to sustainability is resonating through every aspect in the sector. Innovations for cleaner aircraft propulsion to efficient flight operations and from improved aerodynamics to highly efficient air traffic management. This thesis zooms into the latter subject, aiming to contribute to a more sustainable air traffic management system, while maintaining the high level of safety and capacity that is found in the operation of today.

Air Traffic Management (ATM) is concerned with managing the air traffic and airspace, such that flights are executed in a safe and efficient manner. This includes many services such as air traffic control, navigation, information and emergency services, but also demand and capacity balancing of the different airspace sectors. Many different parties are concerned within the ATM system, which makes it a very complex environment. Efficient flight execution is therefor dependent on the ATM services, and the European Union [11] estimates that 6% of emissions during the flight are induced by ATM causes. This can for example be caused by detours around airspace, but also includes aircraft loitering before landing or take-off. A lot of research efforts are ongoing to reduce the inefficiencies, amongst which improvements in Demand and Capacity Balancing (DCB), which this research is concerned with.

Demand and capacity balancing is the process of managing the traffic flows through an airspace block such that safe and efficient flight operation can be guaranteed. Where demand is the number of flights inside the sector, and the capacity is the ability of the responsible Air Navigation Service Provider (ANSP) to keep separation between flights and allow throughput. Luchtverkeersleiding Nederland (LVNL), the Dutch ANSP has recently implemented a decision support tool that aids controllers to balance demand and capacity up to five hours ahead. When a traffic overload is expected, the controller can either assign more resources to increase capacity, or constrain the number of flights allowed to enter. This effectively delays aircraft, which results in extra emissions when airborne, or introduces interruptions in the planning for airlines, passengers and airports. The demand predictions are therefor an important shackle in the chain for efficient flight operations. Unfortunately, demand predictions contain errors and have a uncertainty, leading to unnecessary regulations and inefficient resource utilisation. From these observations, the objective of this research is established: To improve air traffic sector demand forecasting in the tactical domain, by exploring machine learning based trajectory prediction.

With this objective in mind, chapter 2 outlines the problem in detail and formulates a research question. Subsequently, chapter 3 explores how demand prediction is applied in current operation and what alternative methods exist. Based on these findings, chapter 4 describes trajectory prediction in similar fashion. The conventional methodology is first explored, after which more recent innovations and data driven trajectory prediction are discussed. Since the objective of this research is to explore machine learning methods, chapter 5 describes the most relevant machine learning algorithms that have been

applied for demand and trajectory prediction purposes. Additionally, promising alternative methods are also discussed in this chapter. Having explored the methodology framework, chapter 6 outlines the available dataset. How this data is processed, and an initial analysis of demand and trajectory prediction error is given in this chapter. Based on these findings, chapter 7 presents a strategy on the development of the model, and what experiments can be considered to answer the research questions. Finally chapter 8 includes the planning for the remainder of the research project before the first conclusions are given in chapter 9.

2

Problem Statement

2.1. Background

This research is performed at the Delft University of Technology (TU Delft), and supported by the Knowledge & Development Centre Mainport Schiphol (KDC). KDC is a research institute in the Netherlands, focused on ATM research and development of Schiphol airport. This organisation was founded by Dutch aviation stakeholders amongst which LVNL, the Dutch ANSP. Moreover, KLM and Amsterdam Airport Schiphol are also involved. One of the latest developments at LVNL is the Capacity Management Decision Support Tool (DST). This DST is a toolkit to assist air traffic control supervisors in planning and allocating resources during the day of operations for flow and capacity management. In the past, supervisors made capacity management decisions based on experience and expert judgement. However, with increasing numbers of traffic and complexity, the solution space becomes less straight forward. To aid supervisors in their decisions, the DST will support supervisors with accurate information and possible solutions to a traffic overload. LVNL[15] specifies the following requirements of the Decision Support Tool:

- Provide more accurate and complete insight into air traffic load.
- Provide more accurate and complete insight into capacity/workload of runways, airspace and air traffic controllers.
- Provide tools to balance capacity and demand.
- Signal possible overload situations.

An important pillar of the DST framework is a reliable forecast of the traffic load, also referred to as demand. Demand in this case means the number of aircraft that occupy an air traffic control sector during a specific time window. The demand must be balanced against the available capacity. The capacity is defined as the maximum number of aircraft that can safely flow through an airspace sector under guidance of the air traffic controller. Capacity can for example be influenced by having multiple controllers active, or splitting the sector into different segments. When demand exceeds capacity, aircraft still on ground can be delayed (regulated). Airborne flights may be stacked into a hold. These measures however are very undesirable. It causes extra fuel consumption, reduced efficiency and network disruptions. Therefor, effective resource allocation to balance capacity and demand is important. This heavily relies on the demand forecast, which must be reliable and stable on a look ahead time of maximum five hours. There are many methods to make demand predictions, which will be further addressed in chapter 3. However, it is important to note that the Dutch situation is predominantly depending on traffic towards Schiphol. Being a large international hub, the arriving and departing traffic flows from Amsterdam are the main traffic flows in the Netherlands. Furthermore, LVNL only manages airspace up to FL245, which constrains the demand forecast to Area Control and Approach sectors. It is therefor that the demand forecast of DST is highly depending on the arrival times of aircraft in these sectors. In its current form, the DST uses arrival times as received from EUROCONTROL.

When LVNL[40] validated the predicted demand with the actual demand, it became clear that errors persist. One of the consequences of faulty demand predictions can be an overestimate of traffic that

exceeds the available capacity. Based on this information, a supervisor is likely to regulate the incoming overload of traffic, which can eventually lead to the delay of flights. Such precautionary measures are necessary to ensure safe operations. However, when the demand was an overestimation, it might have been an unnecessary regulation. Besides imposing unnecessary delay, the regulation can sometimes backfire: Delayed flights will arrive later, which might cause unexpected high traffic loads that disrupt the network again. In short, unreliable demand predictions can lead to inefficient traffic flows.

2.2. Research Objective

With the capacity and demand balancing applications in mind, it is clear that stable and reliable forecasting of air traffic demand in a given sector is very important. Both for the efficiency of the ATM system as well as the safety performance. Current methods for demand prediction within the tactical phase up to five hours in the future provide reasonably accurate results. Supervisors however experience demand errors and uncertainty that are not good enough for reliable regulation yet. As a result, the objective of this thesis is:

To improve air traffic sector demand forecasting in the tactical domain, by exploring machine learning based trajectory prediction.

This goal opens a suite of novelties to the demand prediction problem. The trajectory based approach is chosen to align this research with the general development direction of future ATM system, as envisioned by Single European Sky ATM Research (SESAR) and Next Generation Air Transportation System (NextGen). These developments propose a future ATM system that will rely on the concept of Trajectory Based Operations (TBO). Furthermore, the machine learning methodology is a rapidly developing research field with a wide range of possibilities. This objective allows to find a data driven approach that can add value by capturing underlying patterns in the traffic flows. The objective can be further specified into a research question, which is given below:

To what extent can machine learning be applied to long term trajectory prediction, and how can its output contribute to demand forecasting of a particular air traffic sector?

This question has three primary elements: Demand and capacity balancing, trajectory prediction & machine learning methodology. These elements are worked out in further sub questions that help considerably to structure the research. The sub-questions are defined as follows:

1. What data features contribute most to the demand prediction accuracy of an air traffic sector?
2. Which methods and best practises are currently applied to forecast the demand of an air traffic sector?
3. To what extend can demand and capacity balancing be stabilised and improved via a data driven demand forecasting approach?
4. Which input parameters are available pre-departure, and which features are expected to contribute most to data driven trajectory prediction?
5. What existing methods have been applied to trajectory prediction?
6. What generative machine learning method, derived from literature, is most suitable to construct aircraft trajectories before departure.
7. What are the benefits of using trajectories for air traffic demand forecasting in an airspace sector that is predominantly based on arriving traffic complexity?

3

Demand Forecasting

3.1. Introduction

To ensure safe flight execution, it is very important that aircraft are separated sufficiently such that no collisions occur. Within commercial aviation, this task is primarily in the hands of ANSPs that guide aircraft along to their destination. To ensure enough separation it is very important that limits exist that prevent an overload of traffic within the blocks of airspace that the skies are made of. This means that ANSPs must set limits that balance between the capabilities of the human operators to maintain high levels of safety, as well allowing enough throughput of traffic for practical reasons. These limitations are called airspace capacity. The next step is then to allow actual scheduled traffic to transit the airspace block. The traffic within this sector is called the demand. Too much traffic within the airspace block is called a traffic overload. This may result in increased probabilities of conflicts. To mitigate this, part of the traffic is often kept in a holding pattern to allow the Air Traffic Controller (ATCo) to safely guide the traffic. As a result the holding flights are delayed and burn additional fuel. This is undesired in many aspects, as it deteriorates safety, emissions, economics and efficiency. Therefore, ANSPs apply demand and capacity balancing to mitigate traffic overloads. By changing the traffic demand or the capacity limits, the overload situation can be prevented.

The Dutch ANSP LVNL has recently developed a new tool to improve the DCB process. With a Decision Support Tool, all aspects of demand and capacity balancing are to be improved. Primarily, this tool should provide more accurate traffic demand predictions and better insights on available capacity resources (runways, airspace configuration, workload and ATCos). With this tooling, air traffic controller supervisors can make better decisions. For example when a traffic overload situation is expected at Schiphol, he or she has a variety of possible options: Activate another runway, divide sectors and allow another controller to assist, or regulate the demand. The regulation option is least desired, as it delays the departure of inbound traffic while still on the ground. This is called Air Traffic Flow Management (ATFM) delay. This is a big problem at Schiphol, which is by far the leading European airport in total ATFM delay, according to research done by Post [46]. Nevertheless, regulations do not always solve the problem. Schiphol is a hub airport, meaning a lot of passengers transfer. As a result, flights that are regulated will still try to arrive as soon as possible to ensure a successful connection for the transit passengers. The pilot will speed up and may still arrive earlier. This can result in a traffic overload in spite of the regulatory mitigation.

As mentioned in the problem statement, in this study only demand forecasting will be considered. Currently most ANSPs make their own demand predictions. The conventional methodology is the trajectory based approach where flight plans or flight schedules are used to estimate a trajectory. From this, expected sector occupancy information can be derived. ANSPs that are joined by network management initiatives such as by Eurocontrol in Europe or the Federal Aviation Administration (FAA) in the US can rely on larger demand prediction systems. Examples of this are the Eurcontrol ETFMS & PREDICT system, or the FAA TFMS system. This trajectory based approach will be explained in section 3.2. In more recent academic research, different approaches to demand forecast are being explored, such as sector flow models. This method aggregates flights and looks at the entire system of flights, rather

than individual flights. The interaction between sectors changes the demand parameters. This is further explained in section 3.3.

3.2. Trajectory Based Demand Prediction

3.2.1. Conventional demand prediction

Classical demand prediction is based on trajectories. Trajectories are sequences of Four Dimensional (4D) aircraft positions. Knowing the aircraft position and altitude at a specified time, it can be evaluated if the aircraft is within an airspace sector. Combining information from all flights gives the total aircraft count in a sector at each moment in time. In most studies, this is the definition of demand. However, some studies deviate from this definition. For example Wang et al.[24] consider the complexity of the traffic within the sector as demand. An organised flow of traffic that follows airways with a common direction on non-conflicting altitudes is less complex than a flow with a lot of converging flight paths. However in most current applications this is left to supervisor judgement for determining capacity. The advantage of a trajectory based approach is that the entire flight profile is available. This allows traffic dynamics and complexity evaluations such as proposed by Delahay et al. [13] or Wang et al. [24]. These studies shows that complexity metrics such as trajectory changes (speed, heading and altitude), conflicts and traffic count can be combined to give a complexity index. With this index, capacity resources can be allocated more efficiently compared to peak traffic count as the demand metric. This has a great potential to increase sector capacity and efficiency while maintaining high safety standards.

In some applications the demand is simplified to the arrival flow only. This flow is defined as the number of aircraft entering the sector during a given time window. Ma et al. [35] modelled sector entry flow based on flight trajectories and trained a graph neural network to predict entry flows on a 2 hour prediction horizon for the French airspace. This may at first seem to be a good parameter for demand and capacity balancing, but this simplification cannot always be applied. The correlation with demand is very dependent on the traffic characteristics. For example, an Upper Area Control centre such as Maastricht Upper Area Control (MUAC) usually sees a variety of flows. There is a large bunch of east-west traffic from the United Kingdom and Oceanic areas, as well as a large bunch of flights from Dutch, German and Belgian cities to all directions. As a result, the sector occupancy time for MUAC sectors has a relatively high variance, as was analysed by Könneman [26]. This means that the arrival times within the sector do not provide enough information to predict demand, hence this assumption cannot be made. For approach sectors, the entry times can sometimes be used as good indicator for demand. The flows are generally organised because all traffic converges towards or diverges from a runway. Fixed routes such as a Standard Instrument Departure (SID) and Standard Arrival Route (STAR) are organising traffic flows, which can make the throughput more constant. For such applications the arrival or entry time can be a good indicator of demand.

Current applications

As mentioned before, most of the current demand forecast tools in operation rely on the sector entry and exit times as predicted by the Eurocontrol ETFMS system or the FAA TFMS system. These flow management systems currently calculate the sector crossing times with an aircraft performance based trajectory predictor. A detailed explanation of this methodology is given in section 4.1. In short, the flight plan is received from the Airspace User (AU), which amongst other information contains the proposed route, departure time, cruise speed, cruise flight level and aircraft type (ICAO [25]). The Trajectory Predictor (TP) matches the expected performance (e.g. BADA model) of the aircraft to the proposed plan. Also other information is added such as atmospheric data. Then the model computes a deterministic 4D trajectory. In a model based approach this is usually done separately for each flight phase: Climb, descend & cruise. With this 4D trajectory it can quickly be evaluated when the aircraft is in each sector. These estimated timings are the final output which is available in the Network Manager Business-to-Business (B2B) dataset (see: section 6.1) that this research makes us of. Also part of the output is a new flightplan that is enriched with time and altitude information. All waypoints are given an estimated overhead time and estimated overhead flight level. For demand calculations however, ANSPs conventionally use the estimated airspace entry and exit times.

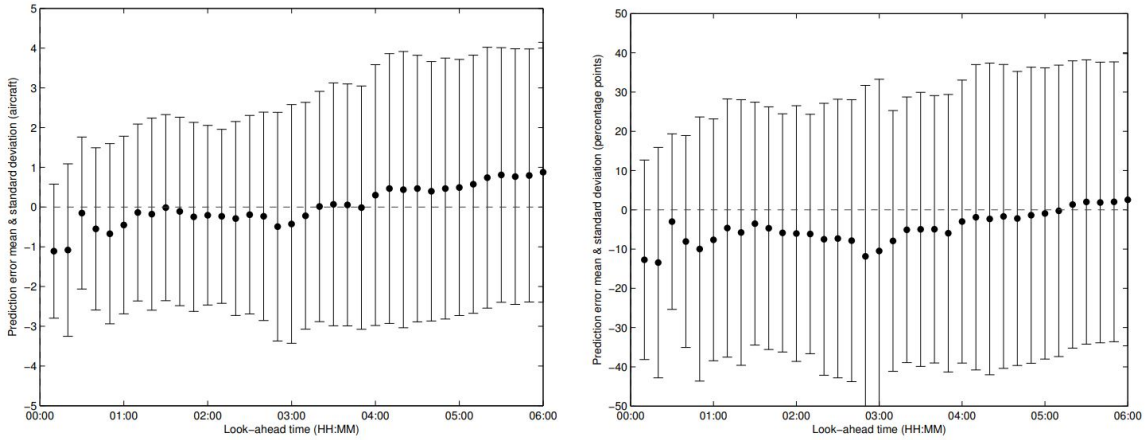
Uncertainty

The actual TP model that provides this information is not publicly available. This is inconvenient as it cannot be easily analysed how demand prediction errors are conceived. There have been studies that thoroughly analyse demand forecasts based on the information provided by a flow manager such as the Eurocontrol Network Manager (NM). Könnemann [26] extensively analysed this data to quantify the uncertainty of the demand predictions for MUAC upper area control sectors in the Benelux and Germany. The study shows what factors contribute to demand uncertainty, the results are displayed in Table 3.1. Looking at the error sources, the first two items are: Departure Time Prediction, and Prediction of ATFCM & ATC interactions. Departure times are mostly influenced by; either a result of cumulative delay throughout the day, or based on the ground processes. For this study, departure time errors will remain out of scope. ATC-ATFCM interactions on the other hand are largely airborne based and therefore will be within the scope of this study. A data driven trajectory prediction approach can be trained to recognise patterns due to controller intervention. The following items (3-5) all lead back to trajectory prediction inaccuracy. Horizontal and vertical route errors are directly obtained from the trajectory predictor, and flight speed is the first derivative thereof. From this review it can therefor be concluded that a significant portion of demand error leads back to the trajectory predictor. Depending on the chosen methodology, a lot of improvements can be made that extend further than pure trajectory prediction error.

Table 3.1: Factors affecting sector demand predictions, as described by Könnemann [26].

Remarks	Factors	Examples
Error sources sorted by level of influence on sector demand prediction, high to low	1. Departure time prediction 2. Prediction of ATFCM initiatives and ATC actions 3. Horizontal route prediction accuracy 4. Vertical route prediction accuracy 5. Flight speed prediction accuracy 6. Changing airspace adaptation data 7. Weather and wind forecast accuracy 8. Surveillance data accuracy 9. Flight technical and operational errors 10. Trajectory models accuracy	1. Abnormal surface events, unavailable gates 2. Non-standard procedures, style and preference of controllers
Factors affecting sector demand prediction	1. Airspace 2. Time 3. Prediction 4. Weather	1. Sector altitude, class, traffic type, primary traffic type (departures, arrivals, en-route, mixed), ACC 2. Day of week, hour of day, time of year (season) 3. Look ahead time, peak count 4. Severe weather, jet-stream
Investigated factors affecting sector demand predictions	1. Weather 2. Flight plan submission 3. Regulation 4. Flight type	1. Good or bad 2. Scheduled or Filed Flight Plan 3. Filed or Regulated departure times 4. Commercial, GA, or military
Factors affecting FIR demand predictions	Traffic type	Arriving or departing

Although the study of Könnemann is very specific for the local MUAC situation, the modelled uncertainty does represent the order of magnitude of current demand prediction tooling. An example of this is given below where the uncertainty of predictions for MUAC sector Koksy High are shown. Figure 3.1 shows that the prediction error standard deviation is generally around 20%-30%, with a slight average under-prediction. This shows that there is a lot of room for improvement. A supervisor cannot make effective decisions on demand predictions that have high variance. Könnemann argues that the average over-prediction of traffic is likely a result due to outflow traffic: Traffic that was expected to cross the sector, but in the end did not pass through. One of the shortcomings of a conventional model based trajectory predictor is that it cannot easily predict another route than the input flight plan. Outflow can be due to re-routing or directs issued by an ATCo. This is not straightforwardly implemented in a kinetic aircraft performance model. Data driven demand forecasting or trajectory prediction methods may be able to capture outflow.



(a) Aircraft prediction error. (b) Percentage prediction error.

Figure 3.1: Mean & standard deviation sector occupancy count prediction error. Dot is mean, 1/2 bar length is 1 standard deviation. Sector Koksy High. [26]

3.2.2. Improvements and alternatives

In order to deal with the high levels of uncertainty in demand predictions, Gilbo and Smith [21] tried a regression model approach to reduce the variance. As both the mean and variance of the predictions show to be relatively stable, Gilbo et al. believed this can be modelled with a probabilistic approach. Their analysis on the US focused Ehanced Traffic Management System (ETMS) shows high variance of demand prediction error similar to Könnemann. The demand predictions are deterministic, meaning the supervisor will only see a single prediction. Gilbo et al. point out that no information on the accuracy of the prediction is available in the decision making process. A relatively straightforward regression model is applied that considers the predicted demand for the current time window, but also the window 15 minutes before and after. This is because delayed aircraft may shift from one window to the next. The prediction error variance could be reduced by about 0.5 aircraft, as shown in Figure 3.2. The reductions of course depend on the sector and look-ahead time, but in essence it shows that a probabilistic approach can perform better than the deterministic model.

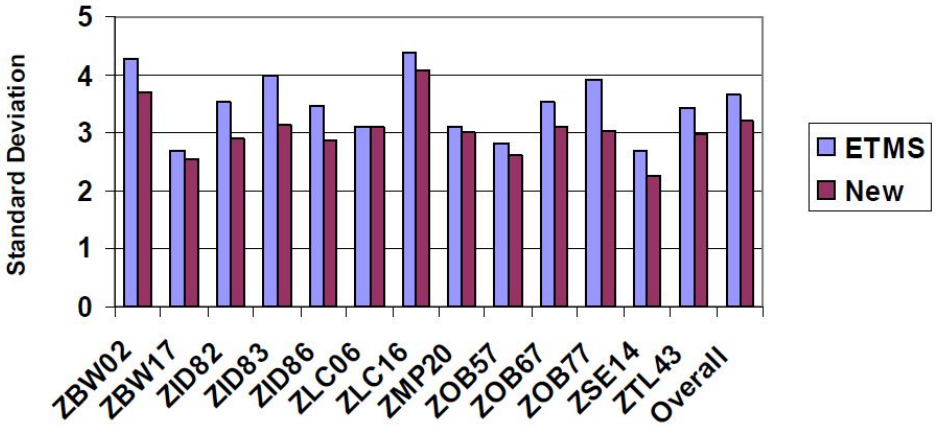


Figure 3.2: Comparison of the demand prediction standard deviation for the conventional ETMS system, and the regression model as developed by Gilbo & Smith [21]. The results are shown per US air traffic sector.

Although Gilbo et al. show improvements on predicted demand with a correction model, another possibility is to improve trajectory prediction. As explained in Table 3.1, the trajectory prediction errors take a significant share of the total error. With this in mind, the SESAR initiative launched a study on improved demand and capacity balancing, which was performed by Fernández et al.[17] and is titled Data-driven Aircraft Trajectory Prediction Research (DART). An important objective of this study is to apply Machine-Learning Approach to Trajectory Prediction, which will be the basis of demand and

capacity balancing. This study focuses on the pre-tactical domain (day -6 to day -1 before operations). The first step is to convert flight plans to 4D trajectories. This is done with a Hidden Markov Model, and will be explained in more detail in subsection 4.2.3. The next step is then to balance the demand to meet the available airspace capacity. This means that delays are sometimes required. The study proposes an agent-based collaborative learning model to calculate and optimise the demand. More specifically a collaborative multi-agent Markov decision problem is applied where trajectories are the agents. Each agent is given a local strategy, policy (or constraints) and rewards if the joint strategy is fulfilled. In this study trajectories can only be delayed, but in a future iteration also 4D solutions can be implemented. Three different collaborative reinforcement learning methods are applied, as convergence is not guaranteed. To experiment with the optimiser, a day in January 2016 in Spanish airspace is modelled. Without any balancing measures applied, the peak occupation count exceeds the available capacity (20) at numerous occasions. This is represented in Figure 3.3 (a). When the agent-based model has optimised the situation by imposing delays, the capacity is never exceeded, as shown in Figure 3.3 (b). However, the error of the new demand prediction and the optimisation have not been validated. This makes it impossible to quantify the performance gains of this machine learning approach. However, this exploratory research is currently still ongoing.

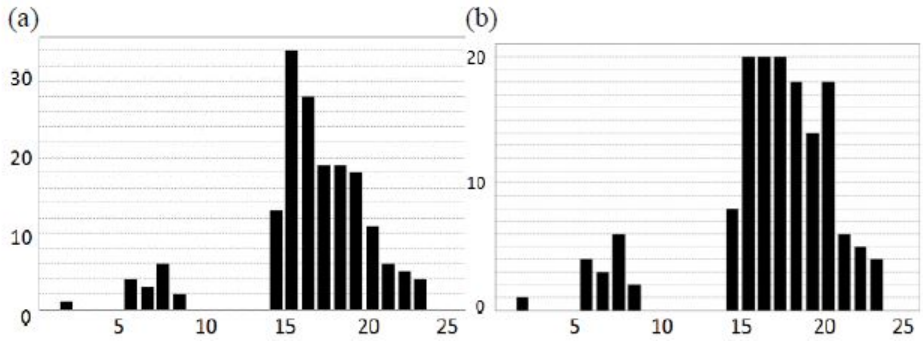


Figure 3.3: Results of the Demand & Capacity Balancing study of the research by Fernández et al.[17]. This figure shows the distribution of interacting flights in Occupancy Counting Periods: (a) initial predictions, (b) after optimisation. The sector's capacity is 20.

The study by Fernández et al. shows that demand prediction and the optimisation thereof go hand in hand. This is confirmed by Xu, Prats and Delahaye [65], that take the trajectory approach one step further. This study compares four optimisation models, where each subsequent model is given more variables to vary with. The simplest model can only impose delay, similar to the DART model. The second model has the ability to select a different trajectory from a set of alternatives submitted by the AU. The last two models can apply different airspace sectorisation and configurations. The cost function is then defined as a composite with delay, ATC operating cost, and trajectory alternatives. The experiment evaluates a full day in February 2017 and comprises the whole French airspace. Trajectories are created with the trajectory predictor by Dalmau et al.[8]. The results show clearly that the addition of alternative trajectories gives the greatest performance gain. This means that less delays can be applied while not exceeding the available sector capacity. Airspace sectorisation and configuration also show to have a positive effect, but to a lesser extend. These outcomes reinforce the general direction of ATM developments: Trajectory based operations have great potential, already in the pre-tactical domain. The challenge is to increase trajectory predictor accuracy, and to introduce a framework in which alternative trajectories can quickly be negotiated between an airspace user and an air traffic flow manager.

3.3. Aggregate Based Demand Prediction

Traditional demand forecasting methods were primarily based on trajectories derived from flight plans. While trajectory prediction on short look ahead times for airborne flights are very accurate, this is less so the case for longer look ahead times. Air Traffic Control (ATC) interactions, weather, or even ground processes when the aircraft is not yet airborne, introduce large discrepancies between an actual and a predicted trajectory. While some efforts have been put in TP and DCB improvements (see subsec-

tion 3.2.2), a lot of research has been done on a different concept: aggregate demand forecasting. This concept does not consider individual trajectories, but rather looks at the traffic flows running through the sector and adjacent sectors. There is a large variety of these aggregate flow models, which will be discussed below.

3.3.1. Sector flow models

One of the first models to step away from trajectories in demand forecasting was presented by Sridhar et al. [49]. This research models the US National Airspace System with 22 airspace blocks, and an international block. A schematic overview of such an airspace block is given in Figure 3.4. Aiming to predict aircraft occupation count as demand, the selected model is a linear dynamic system. This state space system requires an input dataset containing takeoff and landings in each sector at each time step. The state of a block represents the current aircraft count. With a transition matrix that explains the flow of traffic between sectors, the state of the next time-step can be calculated. The transition matrix is calculated based on the observed probabilities of traffic flow between sectors in the training dataset. For each hour of the day a new transition matrix is calculated.

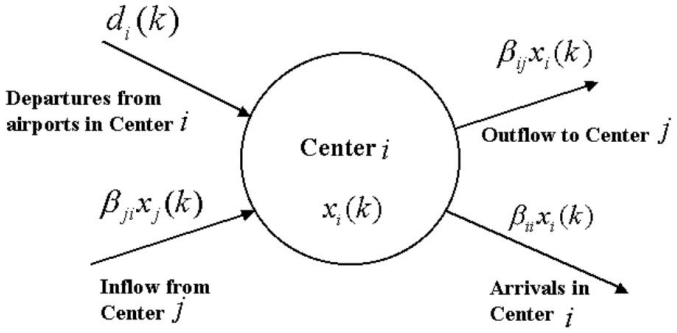


Figure 3.4: The components of aircraft flow contributing to the traffic count in a given centre, as described by Sridhar et al. [49].

The experiment to find the performance of the flow model was to first train the system on two consecutive days of traffic. Then, the third day was modelled with solely the flight schedule as input. The demand curves for most centres show a slight delay with respect to the actual traffic numbers, and an under prediction of the peak moments. However, Sridhar et al. argue that the peaks could still be discerned and therefor the system can be used to alert a demand overload situation. The main benefits of using an aggregate model are to be found in the complexity and magnitude of the calculations. In this specific case, only 23 airspace blocks are present, which is a lot lower in dimension than predicting an equivalent of 5000 trajectories. The authors state that a flow model is less susceptible to uncertainties and that a wide variety of system engineering tools can be applied to improve the model.

In a follow up study, Sridhar et al. [50] aim to reduce the prediction errors by training a set of different models. These models are trained on different datasets to include daily, weekly and seasonal changes. The predictions are compared to the actual demand after which the error is forwarded to a probabilistic hypothesis testing block. This block is able to identify the best fit model and can merge the predictions of multiple models if there is no clear favourite. A schematic overview of such a model is shown in Figure 3.5. However the selection via hypothesis testing in this study can only be done knowing the errors. Without information on the error, no probabilities can be assigned. This is therefor not an autonomous prediction model yet, but a simple learning or classification algorithm may be applied in the next iteration. The results of this aggregate sector flow model show a Root Mean Squared Error (RMSE) between 1.79 and 2.64 aircraft for upper area control sectors in Indianapolis with 2h look ahead time.

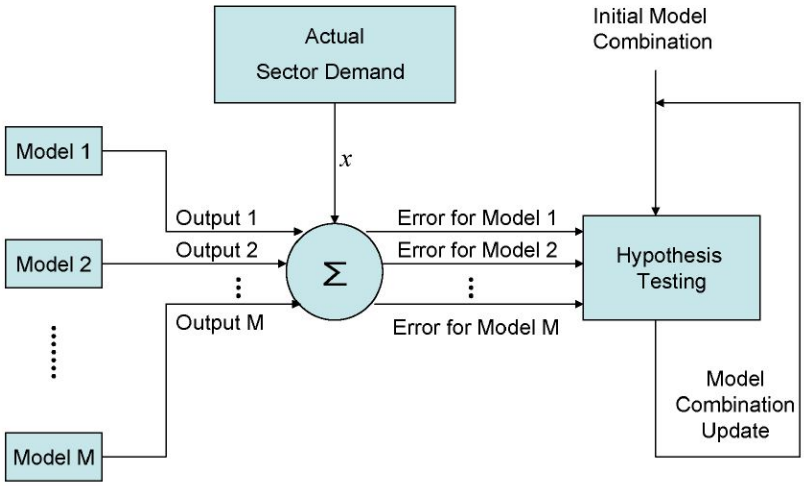


Figure 3.5: Schematic overview of the aggregate sector flow model with hypothesis testing, as proposed by Sridhar et al. [50]

Similarly, Menon et al. [37], base their aggregate model on the LWR road traffic flow theory by Lighthill, Whitham and Richards [31]. This theory states that traffic flows behave like fluids. Flow into the sector affects the density of the sector, and as a result influences the sector output flow. The relation can be described with the differential equation in Equation 3.1, where q is the traffic flow and ρ is the traffic density.

$$\frac{\delta q(x,t)}{\delta x} = \frac{\delta \rho(x,t)}{\delta t} \tag{3.1}$$

Translating this to a block of airspace with a traffic inflow and outflow, and a demand (traffic density), Menon et al. find a linear, discrete-time dynamic system that is very similar to the one proposed by Sridhar et al. [49]. However, the difference is that Menon et al. include an air traffic control action variable that can manipulate the flow through the sector. In the first iteration of the experiment, a scenario is created where five cells are connected, amongst which four airports and one airspace unit. This is shown in Figure 3.6. Each unit has slightly different rules such as inflow/outflow constraints for some airports, and flow control capabilities for different airspaces. The dynamics are then rewritten to a state space system, after which control theory elements can be applied to regulate the flow through the sectors. After checking the controllability of the system, the flow is modelled with controllers in place at the airports. These controllers can limit the flows to ensure steady throughput with respect to the inflow capacity. This automated process was proven to be feasible for the small test environment. Menon et al. call this approach an Eularian approach, compared to the Lagrangian demand prediction methodology which is based on individual trajectories. The study proves that the Eularian approach opens the door to control system implementations that can automate or support demand and capacity balancing.

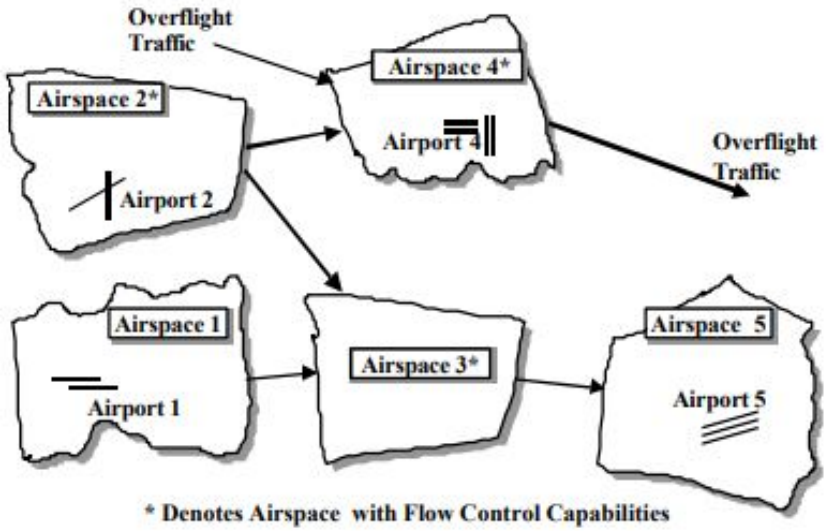


Figure 3.6: The traffic environment used by Menon et al. [37] to design and analyse a traffic flow control system.

The aggregate sector flow model by Menon et. al. showed a lot of potential, but a few problems still exist. One of the biggest constrains is the lack of flow dynamics. For example, aircraft speed is assumed to be constant, and sector transit routes and altitudes are assumed to be constant. In other words, the transition time of traffic through the sector is solely depending on entry or exit flow. To deal with the dynamic spatial and temporal changes of traffic flow observed in actual traffic, Chen et al. [7] improved the Eularian model with different control volumes inside the traffic sectors. The control volumes allow spatial variations (flight paths) and temporal variations (speeds) to model the differences in transit time. Analysing historical radar tracks allows to have these variations within the sector flow model. Secondly, this model updates the transition matrix parameters recursively each prediction step. Chen et al. claim that this increases the model awareness of prediction uncertainty. To test the demand predictions of the model, an upper area control centre of Shanghai, China is evaluated. The demand is predicted with a look-ahead time of 1 hour and for one of the five sectors within this airspace, the results are plotted in Figure 3.7.

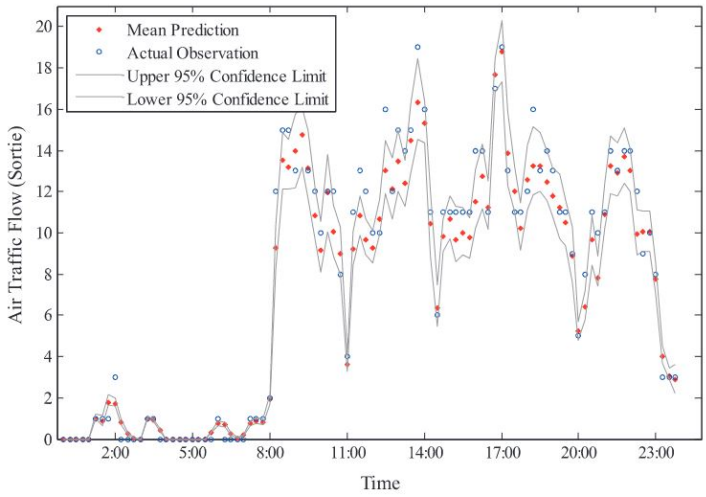


Figure 3.7: Air traffic flow prediction as obtained by Chen et al. [7]. The predictions are the peak flow during a 15 minute window, which is compared to the actual demand of Shanghai upper area centre sector 5.

The errors produced by this model are difficult to compare to the other models, as the conditions are significantly different. The geographical location, look-ahead time, and error metrics are not constant. Nevertheless, in terms of traffic percentage, the mean absolute error is in line with for example the

results of Sridhar et al. [49], and the normalised standard deviation of the error is around 10-15%. This seems an improvement with respect to the analysis of Könnemann on current demand prediction uncertainty, as discussed in subsection 3.2.1.

3.3.2. Alternative models

Aggregate demand prediction is mainly driven by sector flow models, but this is not the only method. In some literature, a probabilistic approach is proposed. Furthermore, over the last years, machine learning applications have also been applied to the demand forecasting domain. In chapter 5, these methods will be thoroughly explained, but the most important findings will be discussed here as well. Lastly, the airport arrival demand also has a lot of commonality with sector demand prediction. This will also be explained in this subsection.

Probabilistic flow model

Stepping away from deterministic to probabilistic demand prediction was already found to be a better method for trajectory based demand prediction according to Gilbo et al. [21]. Meyn [38] took this approach to the aggregate demand forecasting models. He states that when the input data for a predictor model is significantly uncertain, then probabilistic models are superior to deterministic models. With the great uncertainty in aircraft trajectory prediction of conventional demand predictors, this hypothesis may hold for aggregate demand forecasting as well. To test this, Meyn introduced a probabilistic model that assigns a cumulative distribution function to the entry and exit times of the aircraft passing through the sector. This is then evaluated to give a probability of sector occupation to the flight at a moment in time. The difference is then convoluted and aggregated for the entire set of flights, which gives a probabilistic demand. Meyn simulated a 2 hour fictional scenario to compare the model to a deterministic model. The improvements range around 10%-15% decrease of the error standard deviation. The sector demand of a real world scenario is not simulated, so this research is not as far developed as some of the other sector flow models that were discussed.

Random Forest Regression

Besides the probabilistic approach taken by Meyn, another alternative is machine learning. A variety of different machine learning models have been applied to demand forecasting. The first example of this can be found in the LVNL decision support tool [40]. The baseline demand prediction in the DST is purely based on the arrival times provided by the network manager. To deal with uncertainty, a random forest regression model was implemented. The input are the arriving flights communicated by the Network Manager, with arrival times as primary feature. A random forest regression model is trained to predict a new arrival time based on the flight data provided. This model is trained by comparing the predicted arrival time to the actual recorded arrival time. The average error and confidence interval are presented in Figure 3.8. The machine learning model shows significant improvements within 5-hour look ahead time period. As a result, the improved sector entry times lead directly to improved demand forecasts.

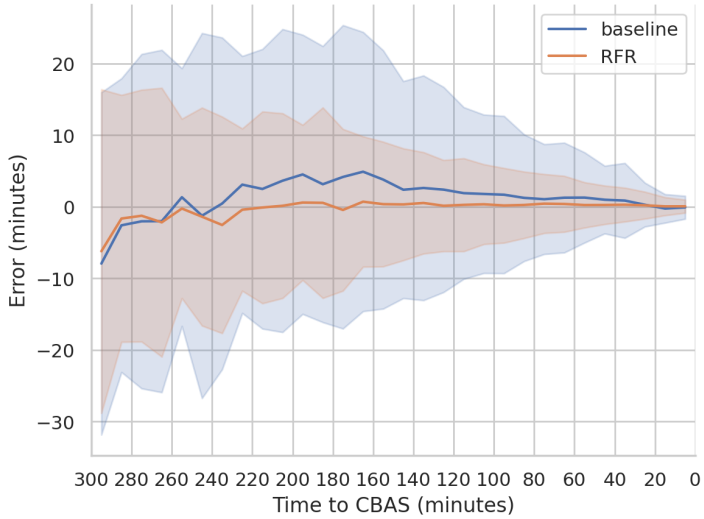


Figure 3.8: The average error and confidence interval of the estimated entry times in the Dutch airspace. The baseline results are the times provided via the Network Manager, and the RFR results are corrected times by the random forest regression model [40].

Neural networks

Neural networks have been amongst the most popular machine learning advancements, and in demand forecasting, this is no exception. The first example of this is the study performed by Lin, Zhang & Liu [32]. Their hypothesis is that traffic demand is the result of spatial and temporal traffic flow. The Convolutional Neural Network has proven to be very effective in retrieving patterns from spatial information such as images. The Recurrent Neural Network (RNN) on the other hand shows excellent time sequence modelling performance. An example of which can be speech recognition. Harnessing the benefits of both models, a hybrid between a Convolutional Neural Network (CNN) and RNN is used to model the traffic flow in the Chinese airspace. The airspace is transformed into a 3 dimensional grid where each cell has a certain amount of traffic: The traffic flow matrix. For each moment in time the flow matrix changes due to flights moving between their origin and destination. The convolutional and recurrent neural networks applied do an excellent job to model the propagation of flows into the future. Hence, the output of the model is an updated flow matrix. The results of the model are compared to other methods such as a flight plan based model and a shallow neural network. The proposed convLSTM network clearly outperforms these models, but it is unclear what look-ahead time the results are based on.

The convolutional and recurrent neural networks are not the only type of neural network that is applied to demand predictions. Another promising method is the graph neural network. The most prominent research on this type of network to the demand prediction problem was developed by Ma et al. [34]. The methodology will be explained in more detail in subsection 5.2.2, but in summary, a network of nodes and edges is made to represent flights that pass through airspaces. This results in a graph network, which is then trained by the message passing principle. Sharing node content information via the edge connections between nodes, at each time-step, allows information to propagate through the network. Ma et al. use this property to set up a spatiotemporal graph network to represent traffic flowing through the air traffic sectors in France. The predictions generated by this network are sector entry flow, which is an important element to construct demand. The experiment is a prediction for an entire day of traffic in France in 2019. The results of a single node are shown in Figure 3.9, meaning this is a specific flow of traffic from sector LFBBP2, via sector LFBBP1, to sector LFBBP3. Combining all the flows through sector LFBBP1 gives the cumulative entry flow. The results of the model predictions on the validation dataset show a considerable improvement of prediction compared to a LSTM neural network, especially at longer look-ahead times.

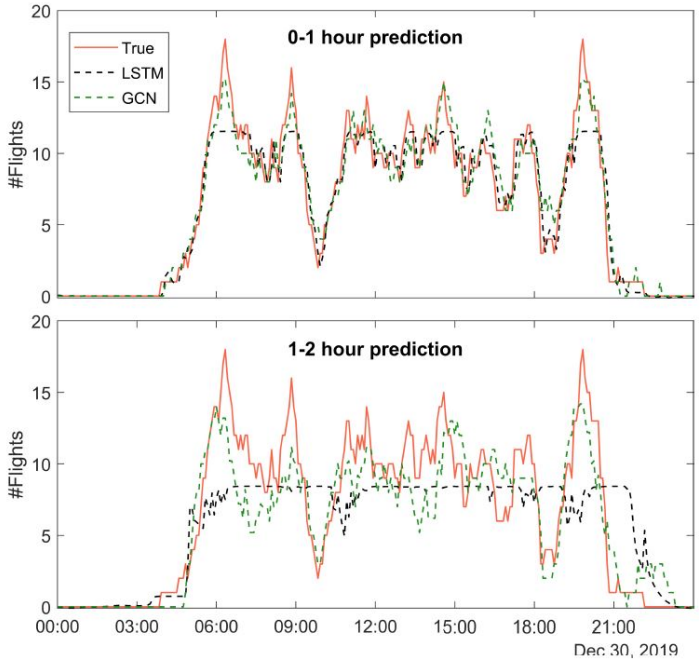


Figure 3.9: Flow prediction result on node “LFBBP2-LFBBP1- LFBBP3” for an entire day in France, as found by Ma et al.[34]. The upper plot shows results for 0-1h look-ahead time and the lower plot for 1-2h look-ahead time.

4

Trajectory Prediction

Having explored air traffic demand forecasting, it was decided to continue developments on a trajectory based model. In this chapter, the Trajectory Predictor will be explained in detail. Trajectory prediction is a task that should result in an accurate estimation of the location of an aircraft in a four dimensional space; latitude, longitude, altitude & time. This chapter investigates the state-of-the-art methods, which can be divided into two categories. The model based approach is worked out in section 4.1, which relies primarily on the dynamics and kinematics of the aircraft in flight. The data driven approach is an increasingly popular method that exploits the large amount of flight data that have become available in recent years. This is explained in section 4.2. After investigating the most relevant methodologies, the prediction accuracy measurements will be discussed in section 4.3.

4.1. Model Based Trajectory Prediction

Model based trajectory prediction can be regarded as the classic method. With a set of input variables, amongst which the state of the aircraft, the environment and perhaps some intent information. A future state of the aircraft is to be predicted with a model that incorporates the physics of the situation. In most TP tasks, the required output is a deterministic 4D trajectory. This includes the 3 spatial dimensions: Altitude, longitude and latitude. The fourth dimension being the time. However, some other states can be included in the prediction as well. to name a few, the speed, mass, heading or thrust can be relevant states for some applications. In ATM however, the 4D trajectory is the core of the future Trajectory Based Operations (TBO) concept. An accurate 4D trajectory can provide sufficient information to do conflict detection & resolution, arrival management, or Air Traffic Flow Management. The accuracy however is very much dependent on the input data, as well as the capabilities of the model. In Figure 4.1, a schematic overview is given of the model based TP components. The following sections will explain these elements while providing examples of recent academic work.

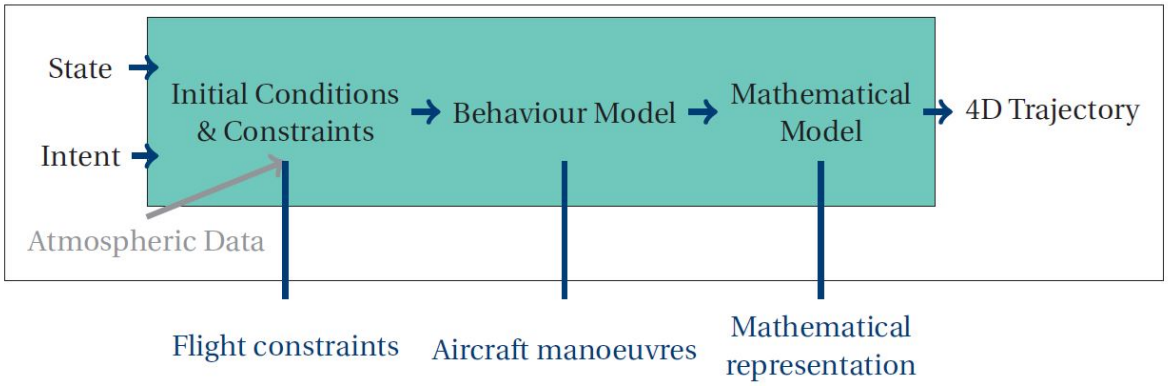


Figure 4.1: Simplified representation of TP concepts, as explained by Tielrooij [54].

4.1.1. State, Intent & Atmospheric Data

State

The state of the aircraft before prediction in a model based approach can be just the current position, altitude and time. For a true kinetic model, the state of the next time-step will be based on the displacement of the aircraft. Newtonian physics require equations of motions. Better estimates can be produced with better input data and less assumptions. Hence, information on the aircraft mass, thrust, configuration (e.g. Flaps, gear), control inputs and many more, can all be included to make improved calculations. The difference between a very simple kinetic model and a complex one often differentiate itself with the inclusion of more state parameters.

Intent

The aircraft intent is the state that the aircraft aims for at a future time-step. Zooming out to the entire flight, this is usually to arrive at the planned destination on the scheduled time. Eurocontrol requires that at least 3 hours before the flight departs, a flight plan is filed that further describes the intended route [3]. During the flight, the intent can be a certain cruise level, speed, waypoint or a heading. Moreover, the intent can include a procedure that the pilot programmed into the Flight Management System (FMS). All this information contributes to what the aircraft is aiming for. Of course, intent is not always exactly followed and might change. For example bias or noise is introduced in executing the intended trajectory when a pilot changes from automatic to manual control, or when wind gusts affect the speed. The intent is usually a collaborative effort between ATCo instruction, and decisions made by the AU. This means that intent information is distributed among stakeholders of the flight. Initiatives such as System Wide Information Management (SWIM), Automatic Dependent Surveillance - Contract (ADS-C) common server, & Essential Ground-Ground Interoperability ((e)IOP) should make intent information more readily available. But as Tielrooij et al.[55] point out; this is still a work in progress.

Many studies have aimed to include rich intent information to the TP, to reduce prediction uncertainty with great success. For example Konyak et al. [27] have tested how ground based TP uncertainty for the descend phase can be reduced by including FMS intent information received from the flight deck. The uncertainty in both Top-of-Descend location and lateral profile could be reduced significantly when intent in the AIDL format was provided to the TP. To illustrate, the lateral prediction root mean squared error could be reduced from 18 nautical miles to just 0.3 nautical miles at the end of the descend phase. This is a massive improvement in TP accuracy. However, the research was very clinical with just one flight and with highly detailed intent information. For example, every configuration and aircraft state change was listed. But it shows the potential of including very detailed intent to existing trajectory predictors. Another example of intent inclusion was performed by Tran et al. [57]. From actual cruise flight data, waypoints were matched to create an intended profile. Although the methodology for this study was data-driven, it still showed that significant improvements can be expected with the introduction of the aircraft intent, other than just the flight plan.

Atmospheric Data

Lastly, the aircraft trajectory is very dependent on the medium that it is moving through. According to Cole et al.[9], wind, temperature, clouds and precipitation can influence the trajectory and the predictability significantly. Any information about the weather along the route can improve predictability. Numerous TP's have included atmospheric data in the predictions. This can simply be the inclusion of wind to obtain ground speed from True Air Speed (TAS). Other predictors also use temperature and pressure information for Indicated Air Speed (IAS) and altitude calculations. This data is of high value to predictions at short look ahead times. For longer look ahead times, the introduction of convective weather can increase predictability. Convective weather can be anticipated on medium term look ahead times, which may explain re-routings. For kinetic models however, this is not straightforward to capture in a physical model, as the re-routes are human decisions mainly. However, the inclusion of convective weather can be very relevant for data driven methods as was demonstrated by Liu et al.[33]. This will be discussed in section 4.2.

4.1.2. Initial conditions & constraints

The aircraft state, intent and atmospheric data input can be regarded as the initial conditions for the trajectory predictor. However, this information is not always readily available for the trajectory prediction

task. Assumptions might be needed to initialise the model, depending on the specific application. For example, consider a flight on an oceanic crossing where position data is not always readily available. Another example can be, that an arriving flight may not yet have a landing runway assigned. This means a lack of intent information. As a result, a TP may need to make assumptions in order to predict the trajectory up till the landing. Hence, the initial conditions are primarily determined by the availability of data.

Also, constraints must be considered before running the model. This can be as trivial as including the constraints of the flight envelope for different aircraft types. This can include minimum and maximum speeds for climb, cruise and descend, but also maximum cruise altitudes. Other constraints can be to include rules about procedures such as overflying waypoints, altitude and speed limits. Also airspace imposes constraints via different airspace classes, prohibited zones and other varieties. This is only to name a few constraints that may be relevant for a trajectory prediction task. It is foremost the application of the TP that determines what constraints must be included: For arrival management it is very important to include constraints about runway use, whereas for Conflict Detection And Resolution (CDR) the performance constraints may be much more relevant.

4.1.3. Behavioural model

The behavioural model can be introduced to explain differences in flight operations as observed between aircraft or even airlines. Tielrooij [54] states the following: ”The constraints model describes what trajectory will be flown, the behaviour model describes how an aircraft will manoeuvre to achieve that trajectory”. This includes bank angles, reaction time after an ATC instruction, transition between cruise, climb & descend, configuration changes and flown speeds. These elements can be regarded as an additional intent that contributes to the future trajectory. However, the behavioural model is often based on observations and practises rather than a direct input as the primary intent parameters. For long look ahead times, the behavioural model is a less significant contributor to model fidelity and will therefor remain out of scope in this research.

4.1.4. Mathematical model

At the heart of the TP is the mathematical model. This element largely determines the accuracy of the prediction, by including input or making assumptions. The mathematical model is also what sets this method apart from the less conventional data driven approach (see section 4.2). Generally speaking, two methods can be inferred from literature.

Kinematic model

The *kinematic* approach does not model all the forces on the aircraft, but rather directly calculates the motion of the body. For ATM purposes this is mostly a point mass with kinematic properties. The point mass assumption can be made because yaw, pitch and roll motions have less relevance for ATM tasks such as conflict detection. An example of a kinematic model can be the WRAP performance model as part of the OpenAP model designed by Sun [51]. This aircraft performance model is publicly available and was build upon Automatic Dependent Surveillance - Broadcast (ADS-B) measurement data. OpenAP consists of a kinematic model, thrust & fuel flow model, and a drag polar model. Although not yet applied in operational trajectory prediction systems, it may very well be capable to perform such tasks.

Kinetic model

The kinetic approach is a Newtonian model that applies forces on a body and calculates the resulting translations and rotations. For ATM purposes, mostly Point Mass Models are exploited, meaning that yaw, pitch & roll motions are ignored. Examples of the kinetic approach are numerous, but most famous is the Base of Aircraft Data (BADA) model as developed by Eurocontrol. Several studies have used and shown the applicability of BADA to ATM research. Poles et al. [45] have analysed and compared the performance of two different BADA models over the entire flight envelope. Gallo et al. [18] have shown how the BADA aircraft performance model can be used in a trajectory predictor. Kinetic models such as BADA model the forces that act on the aircraft centre of gravity. Those being the aerodynamic forces Lift (L) & Drag (D), as well as the propulsive force Thrust (T) and the Weight (W). The total energy model is then used to predict the motion of the aircraft, which is shown in Equation 4.1. In this

equation, V represents TAS. $\frac{dh}{dt}$ is the time derivative of altitude, meaning the vertical speed. Lastly m is the aircraft mass.

$$(T - D)V = W\frac{dh}{dt} + mV\frac{dV}{dt} \tag{4.1}$$

For conventional aircraft the mass changes due to fuel burn. This is accounted for by applying fuel consumption values in the model. Equation 4.1 is the core differential equation that can be numerically integrated to make a prediction of the aircraft state at a future time interval.

For both the kinetic and kinematic approaches, a weather model can be included to make more accurate predictions. Trajectories in ATM are often defined in the geographic reference frame. Aircraft performance however is usually define with respect to the airspeeds. The wind correction from TAS to ground speed is therefor most frequently applied. Other weather influences can be the temperature corrections for altitudes or engine performance. When comparing existing mathematical models such as BADA and OpenAP, there are pros and cons for both methods. BADA is developed and maintained by Eurocontrol, which means it requires a license to use. Sun [51] developed an open alternative with OpenAP. The results of OpenAP and BADA are very similar, meaning both approaches should be capable for model based trajectory prediction.

4.2. Data Driven Trajectory Prediction

Since the digitisation of society and aviation, more and more data has become accessible to researchers in the air traffic management domain. Aircraft have become flying computers and ATM systems have become much more capable for data gathering. As with many other fields of research, large bodies of data can bring new possibilities to existing applications. Namely, accurate measurements of processes can provide statistics and patterns that may have been hidden otherwise. Data driven trajectory prediction models make great use of this by using historic trajectory information as the foundation of a future prediction. First the available data and how to pre-process this will be discussed. Afterwards, examples of the latest data-driven trajectory predictors will be given.

4.2.1. Data review

For data based models, the input data is the most important element to make accurate predictions. Similar to the model based trajectory prediction, aircraft state information, as well as intent, atmospheric data and constraints can all be part of a data driven model. This data comes from different sources, which need to be linked and processed such that the mathematical model can predict the future 4D path of the flight in question. Finding and processing the right sources of data in the ATM environment is not trivial. Data can be proprietary or contain sensitive information. For example the flight data recorder of an aircraft makes highly detailed recordings of all the sensors on board the aircraft. This can be very useful state information for a trajectory predictor. Flight data records are however distributed over a lot of aircraft, meaning a lot of work to gather a significant dataset. Even more constraining is the fact that flight data records are kept confidential by airlines in almost all cases. Furthermore, there is no direct down-link possibility to obtain this information for an online environment. This example shows that a good source of data may not always be feasible. The following paragraphs will discuss examples and best practises of input data for a model based trajectory predictor

Aircraft trajectories

Data-driven TP models are based on actual aircraft trajectory data at their core. This means that at the minimum the 4D state of flights need to be available. Taking the ground based perspective, primary or secondary surveillance radars provide accurate aircraft positional information. Stored radar data can be used to get estimates of the position, altitude and time of aircraft. Alligier et al. [1] use radar tracks for a thrust and mass estimation model, whereas Kun & Wei [28] make an entire radar data based TP. Nonetheless, surveillance radars are not the latest and most accessible source of surveillance data. Alternatively, Automatic Dependent Surveillance - Broadcast is an onboard system that automatically transmits surveillance messages. The core messages contain an aircraft identifier code, position, altitude (both Global Navigation Satellite System (GNSS) & pressure altitude), and speed. Other messages may include aircraft status and capability information [52]. The European Union (EU) has mandated the use of ADS-B from December 2020 [10]. A similar mandate applies to

the US airspace. This means that ADS-B is widely implemented and almost all flights can be tracked with the technology. Besides the high equipage levels among aircraft, it is also relatively simple and inexpensive to set up a receiver unit. Such a unit receives the automatic broadcasts of the aircraft and stores the data. Networks of receivers exist, ensuring a good coverage of the world. Examples of such networks are Flightradar24 and OpenSky. The latter of which is free and open to use for research purposes and has very good coverage in the European Civil Aviation Conference (ECAC) area. Therefore, the ADS-B data provided by OpenSky was chosen as input for this research. Sequencing the position, altitude and speed reports of the ADS-B messages in the network can provide flight trajectories as input to a data driven model.

Aircraft intent

The aircraft intent can be very differently specified in a data driven TP task. From high level flight objectives such as a destination airport or flight plan, to very specific intent such as a slight change in airspeed, or the glideslope intercept altitude. The intent was already explained in detail in subsection 4.1.1. For data-based modelling however, the intent is not only part of the model input specifications, but will also be used to train or build a model. Various data driven studies have used intent information. For example, Tran et al. [57] have made a machine learning trajectory prediction model. They constructed a Convolutional Neural Network encoder and a Recurrent Neural Network decoder with intent as the primary feature. This intent was constructed from ADS-B data entirely in two phases. In the first phase, intent is reconstructed by coupling significantly close waypoints to the flown trajectory. These waypoints will be regarded as planned route. Heading changes in the flown trajectory that could not be explained by waypoints were regarded as ATC instructions, and therefor make another type of intent. The results of the prediction are very good, which is not unexpected when the intent almost completely matches the to be predicted trajectory. Nevertheless it shows the added value of intent for future upgrades of TP methods, once this data is available.

Besides generating intent from ADS-B data, intent can also be constructed from flight plans. Numerous studies use the flight plan as intent feature. For example Liu and Hansen [33] take a generative approach to predict entire trajectories between George Bush International Airport (IAH) to Logan International Airport (BOS). Their LSTM based neural network shows good predictions as a result of flight plan input. As convective weather is a very important element in this study as well, it is hard to quantify the gain of intent, but the final results are a horizontal average absolute error of 50 nautical miles and a vertical error of 2800ft. Using a similar model structure, Rozendaal [47] showed improved route predictions based on flight plan information. The model output are only 2 dimensional routes (longitude, latitude), but analysis shows an improvement over the filed flight plan. This study also includes different departure airports, which shows that intent is not only a local benefit.

Based on the results of Liu et al. & Rozendaal, it can be concluded that the inclusion of the flight plan is a very suitable start to the generative trajectory prediction task. The data source that can be used to provide flightplan data is the Business-to-Business (B2B) feed from the Eurocontrol Network Manager. This data is cordially provided via the KDC and consists of flight update messages. These messages contain the latest flight plan, that can even be updated during the flight. But besides, there is much more data that can be regarded as intent. For example, if the departure airport is a Collaborative Decision Making (CDM) airport, time related information will be shared as well. Delays, regulations, and scheduled timings are all part of this. An extensive description of the available parameters in the B2B dataset is given in section 6.1.

Other intent information can be the list of estimated ATC sectors that the aircraft will transit through. This is based on the Eurocontrol PREDICT system, which calculates when the aircraft will enter and leave each air traffic sector. The B2B data may provide extra context on the intentions of both the AU as well as other stakeholders such as the departure airport or ANSP's. During this study the flightplan will be the primary input, but some of the above mentioned variables may be included in the intent vector.

Lastly, one of the most promising intent data provisions might be the ADS-C connection. This airborne technology can be considered similar to ADS-B, but now on a contractual connection basis with up-link capabilities. This connection will be used to send and receive data between the aircraft and a ground based station such as the ANSP or the airline operation control centre. Regarding intent, Tielrooij et al. [55] argue that sharing the Extended Predicted Profile over the ADS-C connection is highly beneficial

to trajectory based operations. This allows a ground based system to receive a trajectory of up to 144 datapoints that the aircraft FMS is programmed to. This also includes arrival routes, holding patterns or instrument departures. Nevertheless, ADS-C is not widely implemented yet. Currently in Europe, only MUAC has some systems in place to receive ADS-C Extended Projected Profile data.

Meteorological data

Many trajectory prediction studies have shown that the atmospheric conditions play a large role in explaining the variance observed between different trajectories.Data driven models potentially have the ability to find the changes in trajectories as imposed by weather conditions. Ayhan and Samet [4] developed a Hidden Markov Model where weather information was given as a hidden state among the cells of the solution space. These parameters are temperature, wind speed and wind direction. The data was gathered from the National Oceanic and Atmospheric Administration (NOAA) NCES RAP dataset, which has a horizontal resolution of 13km and 50 altitude levels. As a result of this resolution and the HMM capabilities, the horizontal mean error was just below 13km.

Liu and Hansen [33] went one step further and included convective weather (thunderstorms). Their data comes from the North American Mesoscale Forecast system that has 6 hour forecasts of temperature and wind vectors, among a similar spatial resolution as Ayhan et al. The convective weather is obtained from the the National Convective Weather Forecast system. With this extra information, they show that a machine learning generative trajectory predictor is able to predict detours due to convective weather.

Both aforementioned data-sources are common for trajectory prediction studies across the US airspace. However, for the European situation, a different data source needs to be found. De Leege et al. [30] used Meteorological Aerodrome Report (METAR) data for predictions close to the airport. This may however not be a suitable source for predicting entire trajectories, as most of the flight will be outside of the aerodrome region, and METARs are primarily ground based reports. A European counterpart of the NCES RAP data can be provided by the European Centre for Medium-Range Weather Forecasts. Zhang et al. [67] used the wind data from this forecast for their TP application, but at a lower spatial grid resolution of 80 kilometres. This organisation also has a higher resolution dataset available, which may be suitable for use in this research. This is explained in more detail in section 6.3.

4.2.2. Pre-processing trajectories

In many data based trajectory prediction models, some pre-processing choices are made to improve the effectiveness of such models. Most common for machine learning approaches are normalisation and trajectory clustering. Normalisation is a technique that changes numerical values to a bounded scale. This helps to expose distribution or relations in data more efficiently. Clustering helps to group similar trajectories. Specific TP models for each cluster constrain the search space during the training phase. Both methods can increase performance and stability.

Clustering

Clustering is an unsupervised classification method that is very frequently applied in TP tasks to increase the performance of data driven trajectory predictors. For example Graas [23], Ayhan & Hamet [4], Fernández et al. [17], Wu et al. [63], and Wang et al. [59] apply clustering which is shown to greatly enhance the predictability. The idea of clustering is to find trajectories of similar characteristics in horizontal, vertical or temporal profiles. The theory is that different classes of trajectories are the resultant of different conditions. For example different weather conditions, different procedures and airspace, or the season/day/time of flight. The different clusters can each be trained with a dedicated model. Usually the model architecture is the same for each class, but during training the configuration of the model is adapted to each group or trajectories differently. This allows specific features and patterns for each cluster to be captured by the models. K-means clustering and Density-Based Spatial Clustering of Application with Noise (DBSCAN) are the most common methodologies applied to trajectory clustering [42].

Normalisation

Another method to increase model performance is to normalise the input data. Four dimensional trajectory information has different scales and order of magnitudes for location, altitude and time. Normalising this data means that the scale will change. For example, the altitude (feet) is on a completely different

scale compared to lateral/longitudinal changes, which is usually in the order of hundreds of nautical miles. Data driven models do not interpret scales depending on the units, but just perceive numbers. Therefor it may become biased during training and consider a certain data feature with its variance much more important than another. Normalisation helps to scale data accordingly, while not losing any of its meaning. The resulting predictions will be on the normalised scale, but this is entirely reversible to the original scale. There are different methods to normalise, where the application is very much data dependent. Most common are min-max normalisation (Equation 4.2), or z-score normalisation (Equation 4.3). The latter of which was applied by Wu et al. [63] to normalise trajectories for a machine learning based TP.

$$x' = \frac{(x - x_{min})}{(x_{max} - x_{min})} \tag{4.2}$$

$$x' = \frac{x - \mu(x)}{\sigma(x)} \tag{4.3}$$

Another useful method that can be regarded as some sort of normalisation is the cyclical encoding of time. Air traffic has some strong relation with the time of day, days of the week, and season. For example, traffic at Schiphol is arriving and departing in peaks because of the hub function. This makes time a very important element for the demand forecasting purpose of the TP. Time in its original representation (YYY-MM-DD HH:MM:SS) is purely a linear sequence to a machine learning algorithm. For example 23:59 in reality is very close to 00:02, but to a computer it is very far apart. This can be solved by cyclical encoding [40]. Representing the time data (hours, days of the week,and months) by a sine and a cosine, the representation becomes cyclical. Note that both sinusoids are required, as otherwise the representation of for example AM and PM cannot be distinguished. This effectively doubles the input vector which will take extra computational resources. Nevertheless it may be worth the trade-off.

4.2.3. Existing applications of data-driven models

Model based trajectory prediction is not the only type trajectory prediction that is applied. A lot of academic work has recently shifted towards data driven models, because some of the downsides of performance models cannot be overcome within that domain. Especially when the TP is build for longer look ahead times, or generative trajectory prediction, deviations from the intended flight path are more common, whereas intent information becomes less available. Data driven models have the advantage of being trained on historic trajectories, and therefor may be able to capture intent without adding new data. The following subsections will explain some of the data driven methods that have been applied to improve TP performance.

Regression Models

Regression models are very common in many modelling applications. In trajectory prediction however, they are not broadly applied. This is partly because the physics of aircraft motion are known to a large extend, as discussed in section 4.1. Whereas other data-driven models have proven to be more effective. Nevertheless, some studies have applied linear regression to the trajectory prediction problem. For example Hamed et al. [20] have made several models to predict the cruise altitude after climb with a look ahead of 10 minutes. Amongst the set of evaluated models is a linear and Loess regression model that performs significantly better than the BADA point-mass-model. For example BADA reaches an RMSE of 1800 feet, compared to 960 and 910 feet of the linear and Loess regression model.

Another trajectory predictor based on a regression model was build by Leege et al. [30]. To predict the time over waypoints on a fixed arrival route, Leege et al. applied a Generalised Linear Regression Model. This approach is very similar to an ordinary regression model where the output variables have a certain probability distribution. A stepwise approach was applied to evaluate the explanatory power of the input variables. By varying the input variables, the distributions of the output metrics can be analysed to gain understanding of the input effect on the model outcome. This is therefor a supervised training methodology. The mean absolute error after a 45NM route was 18 seconds. However, both methods are only partial trajectory predictors for altitude and time respectively. It is not straightforward to apply regression methods for multiple dependent variables at the same time in trajectory prediction problems.

Hidden Markov Models

Markov models are a popular approach to trajectory prediction with longer look-ahead times. Due to the stochastic behaviour that trajectories may show, the Markov model is generally considered as a suitable method to the TP problem. This method requires a discretised grid which represents the system. The state and transitions of the cells in the grid can then be learned on a probabilistic basis from historic data. This can be applied for many sequential prediction problems, amongst which speech recognition and trajectory prediction. Mostly the HMM is applied, which allows the inclusion of hidden states. Hidden states can for example be atmospheric conditions, or air traffic sector occupancy. The Hidden Markov Model will then be trained to explain transitions based on the patterns and relations within the hidden states. This method is applied by Pan et al. [44], and Ayhan et al. [4]. Both aim to capture the relation of trajectories with weather data as hidden states. Both apply the Viterbi algorithm to train the model, but where Ayhan et al. make predictions of trajectories pre-departure, Pan et al. consider airborne flights during cruise. For both studies, the grid size largely determines the results of the trajectories. The position cannot be determined on higher resolutions than the cells created. The DART research by Fernández et al.[17] also applied a HMM but with demand prediction in mind. The cells in the grid can be made to correspond with air traffic sectors. As a result, the predicted transitions can theoretically be aggregated for the entire set of flights, meaning the output is a demand forecast. A graphical representation is shown in Figure 4.2

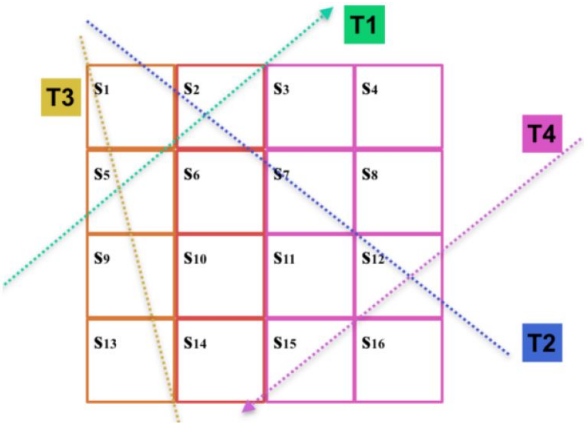


Figure 4.2: Graphical representations of trajectories transiting through blocks of airspace as modelled in the DART study by Fernández et al.[17]. The trajectories can be obtained through a HMM.

As envisioned in the DART study, the demand is based on trajectories, which is very similar to the objective of this study. The discretisation of the HMM can be made to correspond to the airspace sectors. The downside of this method however may be that the resolution of the trajectories is likely to be bound by the size of the airspace blocks. The results of the research by Pan et al. & Ayhan et al. has shown that this is likely the case.

Neural Networks

In recent years, a lot of research efforts in data science have shifted to artificial neural networks. Based on the biological structure of the brain, this method mimics a neural network through one or more layers of nodes. These nodes connect the previous layer to the next and contain a weight and activation function. Such networks can be trained via a process called back-propagation. A cost function calculates the error which is then back-propagated to the network to update the weights of the nodes. Already in 1999, Fablec and Alliot [16] did a first try to implement a neural network for the prediction of cruise altitude. But only since the last couple of years, the application of neural networks in trajectory prediction has seen a surge. Data science fields in which neural networks have proven themselves, such as image classification, are inherently different from the trajectory prediction problem. trajectory prediction is a time sequence modelling task with multiple sequences of features, instead of classification with only one output parameter. This makes it less straightforward to apply a basic artificial neural network in a trajectory prediction task. Wang et al. [59] approached this problem via clustering and principle component analysis as preparation, before applying a simple artificial neural network. The network

receives an input signal, processes the signal in the neurons and feedforward it to the subsequent layer of neurons. This creates a layered network. Mathematically this can be represented by layers of connected nodes which have a certain weight and activation function. The weights can be trained by back-propagating the error via the derivatives of each element. A learning rate then determines by how much the weights will be updated accordingly. A schematic representation of the simple artificial neural network is shown in Figure 4.3, which was implemented by Wang et al. This method proved good trajectory prediction performance on the short term, but longer term predictions stayed out of scope.

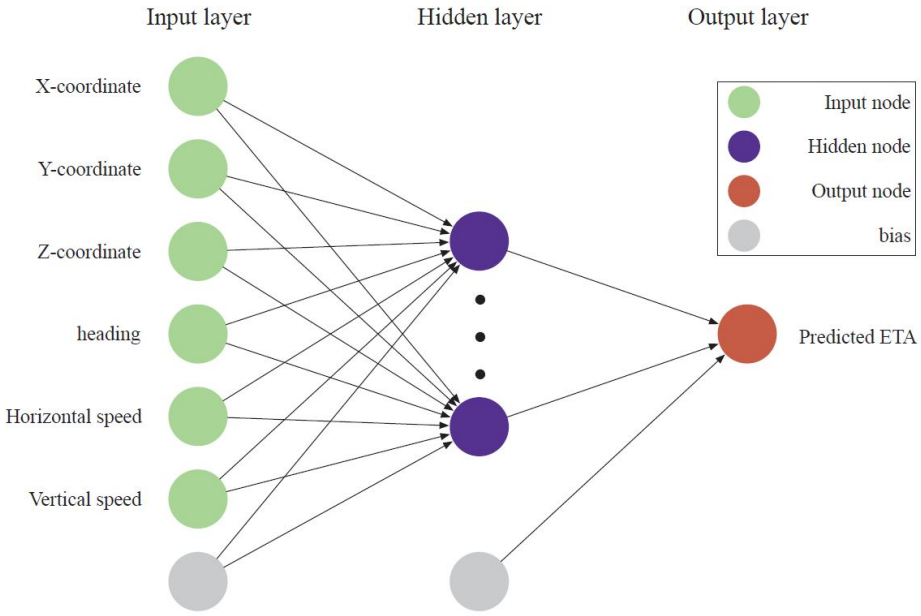


Figure 4.3: The artificial neural network applied by Wang et al.[59].

Another method that became very popular was the use of the Recurrent Neural Network (RNN). These networks have the ability to process time sequences by feeding some of the output back into the network. This network was invented to adapt neural networks to time sequence prediction tasks. Primarily speech recognition and translation tasks were envisioned, but their application to trajectory prediction was found to be an improvement as well. Nevertheless, RNNs were not a perfect solution to the trajectory prediction problem. Large sequence modelling suffers from the vanishing gradient problem, where the useful initial information does not propagate through the network to reach future time-steps. This means that longer look ahead times show exponential increases of prediction error. Also the exploding gradient problem is present in classic RNNs, which means that they may not always converge.

To deal with these problems, a lot of varieties have been created. For example the Long-Short Term Memory Cell (LSTM) cell or the Gated Recurrent Unit (GRU) cell. The working principles will be explained in great detail in section 5.3. As an example, Overkamp [43], Liu & Hansen [33], and Rozendaal [47] all use a LSTM based neural network for the trajectory prediction task. Overkamp applies the LSTM network to predict trajectories up till 20 minutes in a free route airspace, while incorporating traffic dynamics. Rozendaal used the LSTM neural network as a generative model by predicting horizontal routes from the last filed flightplan before departure. A selection of the predicted trajectories by Overkamp and Rozendaal are shown in Figure 4.4.

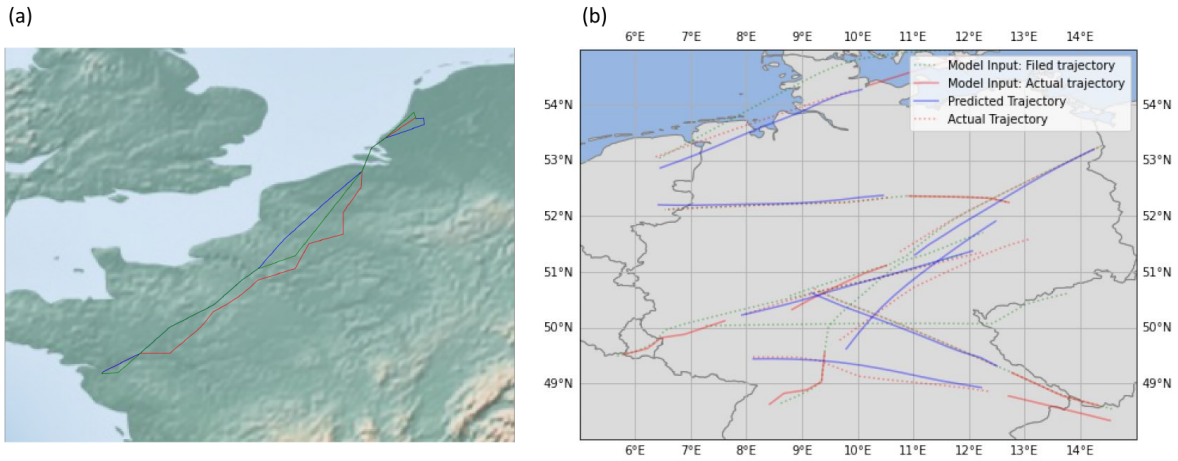


Figure 4.4: A selection of predicted trajectories from the research of (a) Rozendaal [47] & (b) Overkamp [43]. Both apply an LSTM neural network.

Liu & Hansen take it one step further and generate actual trajectories from the flightplan for a single city pair. The encoder-decoder structure combines both CNN and the LSTM based neural networks to also include spatiotemporal weather information. This study shows that neural networks are a feasible method for generative trajectory prediction. Also the GRU network has seen some applications in the trajectory prediction domain. For example, Tran et al. [57] have applied a hybrid CNN & GRU based encoder-decoder model for trajectory prediction with up to ten minutes look ahead time. They include reconstructed intent and show that within the en-route phase, a horizontal prediction error decrease of 30% RMSE can be reached compared to the Eurocontrol standards in CDR tooling.

Ensemble Meta-Estimators

Ensemble Meta-Estimators are supervised machine learning methods that generate a set of many predictions, after which the average or the best suiting prediction is taken as the outcome. This method is primarily used in classification problems, of which random forest and Gradient Boosting Machine (GBM) algorithms are the best example. Both methods consist of individually trained decision trees (forest) and exploit data characteristics to create an ensemble of classifications. Random forests use training data randomisation to train trees for different scenarios. GBM uses weak learners to find what splits reduce the cost function most effectively. Both random forest and GBM have been applied to trajectory prediction tasks successfully. Dek et al. [12] compared GBM to other data-driven TP models such as an LSTM neural network. It was inferred that GBM outperformed the evaluated models, but the look-ahead time was only 120 seconds. This makes the regression sequence fairly short. Because commercial flights usually do not change their motion drastically within 120 seconds, this method resembles a classification problem more than a regression problem. It is expected that other methods perform better on the long look-ahead times that this thesis is concerned with.

4.3. Prediction Accuracy Measurement

Prediction accuracy is the ability of the model to precisely predict the four dimensional trajectory of the aircraft. This can be measured in any of the four dimensions: Lateral, longitudinal, vertical and temporal. Mondoloni et al. [39] defined basic output accuracy metrics that can be used to asses the prediction performance. For Machine learning based models, this is especially important, because the error metrics compose the loss functions for training the model. Observed metrics are:

- **Temporal**: Time error is the difference between the time of a predicted event and the actual time of the event. For an air traffic controller merging two traffic flows onto a route, this can be the time at the metering point. For demand prediction it can be the entry time into the relevant sector. Nevertheless, time is likely to be the running variable during trajectory prediction. Hence temporal error will not be evaluated for training or comparison of trajectories. For demand prediction the temporal error is important.
- **Horizontal**: The lateral & longitudinal error can simply be defined as the difference vector between the predicted and the actual position at a certain timestamp. However, in many TP studies this is further divided into two metrics: The Cross-Track Error (CTE) & the Along-Track Error (ATE). The CTE is the distance of the predicted to the actual position perpendicular to the ground track. The ATE can then be defined as the error distance along the ground track. This is shown graphically in Figure 4.5. For almost any trajectory based tooling, the horizontal location accuracy is very important; be it for conflict detection or sector load balancing.
- **Vertical**: Altitude error is the difference between the predicted vertical position and the actual vertical position at a given moment of time. The vertical accuracy is especially important for tools that require altitude information. For example to find vertical solutions in a conflict resolution tool. The Traffic Collision and Avoidance System (TCAS) is a good example of this.

Besides these basic metrics, the accuracy on secondary states can also be measured. For example speed, heading angle, track angle, amongst others may be taken into account. However, for this study these properties are less relevant. Given that the trajectory is used to make estimations on the demand of the Dutch FIR, the temporal accuracy at the entry point is most relevant. Nevertheless, the vertical and horizontal error metrics will also be evaluated, as these errors will be used in the cost function to optimise the TP module. Also for comparison with other trajectory prediction methods, these errors are important. Considering the TBO approach, the predicted trajectories may be applied to other tools or airspaces as well, so the entire trajectory accuracy is of importance.

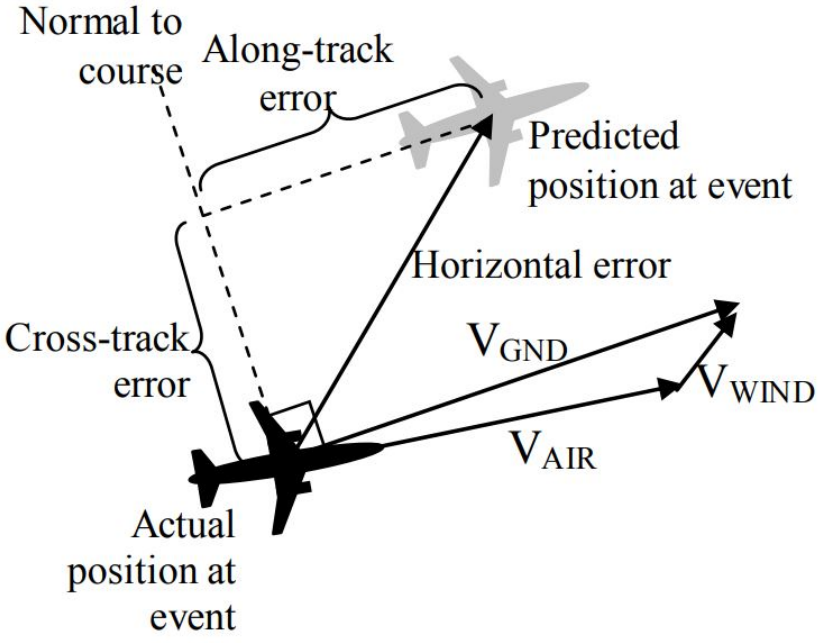


Figure 4.5: Horizontal error metrics for trajectory prediction [39].

Apart from the variables that define accuracy, the errors can be evaluated with different error functions. Frequently used metrics can be the x-percentile method, circular error probable, coefficient of determination, Mean Squared Error (MSE), or Root Mean Squared Error (RMSE). The latter of which is most common in trajectory prediction applications. This metric produces errors with the same unit as the prediction, always has a positive value, and accounts for large errors with respect to the number of datapoints. The RMSE is defined in Equation 4.4. Where y is the actual data-point, $\hat{y}$ the predicted data-point, and n the number of datapoints.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{4.4}$$

Not only the RMSE will be evaluated, but also the coefficient of determination (R^2) may be a good option to evaluate the performance of the prediction. An R^2 value close to 1 means that the TP can accurately describe all the variance observed, and a value of 0 means the opposite. Equation 4.5 describes how to calculate this coefficient. In this equation $\bar{y}_i$ is the mean value of the observed data. Finally, the error variance and standard deviation will be included when reporting the performance of the TP that is build for this research.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \tag{4.5}$$

5

Machine Learning Methodology

Machine learning is a subdomain of artificial intelligence that is devoted to building models that can perform a certain task with learning, or self-improving characteristics. Over the last decades, with the rise of computer sciences and the availability of data storage and processing tools, machine learning has seen lots of developments and applications. As explained in earlier chapters, this is also true for air traffic management. Given the vast amounts of data that are produced in the ATM operations, there are lots of opportunities for machine learning innovations on existing or novel applications. With the research objective in mind, this chapter will go into more detail on what machine learning models are commonly used in demand an trajectory prediction, and tasks alike. Furthermore, models that are potentially relevant to the problem are discussed as well. In section 5.1, a general overview of machine learning is given. In section 5.2, the machine learning applications in demand forecasting are explored. In section 5.3, the models applied to trajectories are discussed. Finally, alternative models that are not yet applied to one of the aforementioned domains are presented in section 5.4.

5.1. General overview

A lot of machine learning algorithms have been developed over the years. The learning aspect of these algorithms are introduced because of various reasons: Sometimes, the solution to a problem is non-trivial and cannot easily be explained with equations or rules. In other cases it is not possible to predict every condition that the variable will be in. For both scenarios, learning can be introduced to solve this problem. Nevertheless, statistical dependencies or correlations must be present in the data in order to have a chance of successful modelling. Generally, machine learning algorithms can be subdivided into three main categories: Supervised learning, unsupervised learning and reinforcement learning. This is shown in Figure 5.1[1].

Unsupervised learning is applied when there is no knowledge on the correctness of the outcome. There is no way to tell the algorithm whether an answer is right or not. What remains is that patterns and groups can be identified. Grouping data on likeliness that the data correspond to a group is called clustering. Dimensionality reduction methods transform the dataset to another reference frame. The information is kept, but patterns may be identified more clearly.

Supervised learning methods can be used when the output labels belonging to the input data are known. The agents can be trained to learn the relation between the input and output pair. Models within this category for example are regression and classification. Fitting multiple models to a dataset and selecting the best for each situation is called an ensemble method. Lastly, deep learning algorithms are introduced for more complex tasks such as sequence modelling, or generative modelling tasks.

Last, reinforcement learning tasks are algorithms in which an agent is rewarded or punished while trying different possible settings and combinations. The rewards and punishments can be designed to lead to an optimum, but allows the algorithm to choose states that cannot easily be predicted otherwise. In this study, reinforcement learning will not be considered: Since the state of both trajectories and

[1] https://thaddeus-segura.com/intro_to_ml/

demand are known and bounded, it is not likely that reinforcement learning is required. Supervised and unsupervised learning methods will be discussed in the following sections.

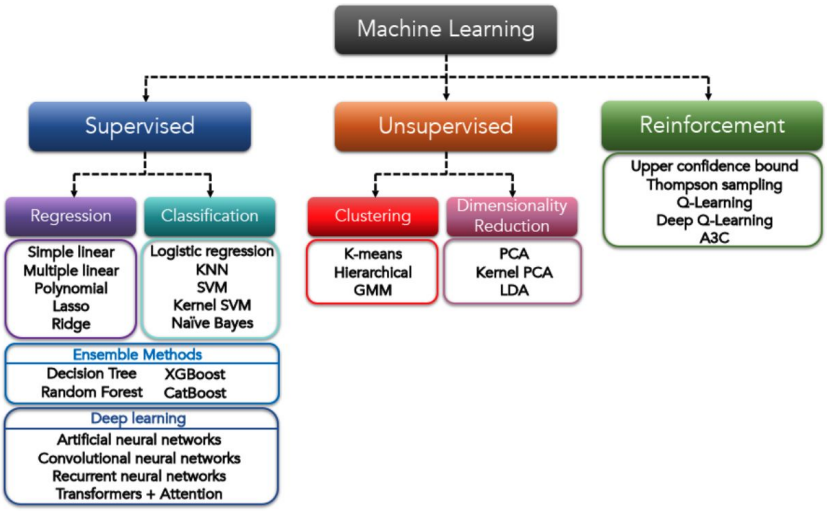


Figure 5.1: Overview of the machine learning categories and respective algorithms.

5.2. Machine Learning in Demand Forecasting

Air traffic demand forecasting has seen a variety of new methodologies that step away from the classical trajectory evaluation method. Conventional systems such PREDICT (Eurocontrol) and the ETFMS (FAA) calculate trajectories and then estimate the demand of air traffic sectors by looking at when flights are inside the sector. Aggregate models disregard trajectories and look at the bigger picture by modelling the total flow of traffic between sectors directly. Predictions of individual flights are not essential to predict the total demand. The aggregate models have been improved a lot with better data and machine learning models. The objective of this thesis is to explore machine learning models that may be valuable to improve demand forecasting. Although lots of models have been tested, in this section the convolutional neural network and the graph neural network will be discussed.

5.2.1. Convolutional Neural Network

Aggregate demand forecasting models generally evaluate larger airspace systems instead of a single sector. By finding patterns in the flow between air traffic sectors, a model can be build to predict propagation of the flow. This system identification task is very suitable for neural networks. In this subsection the convolutional neural network will be explained on the basis of two air traffic flow models. Xie et al. [64] use a convolutional neural network to evaluate traffic complexity in a sector, and Lin et al. [32] use a CNN in combination with a recurrent neural network to predict traffic flow propagation.

Convolutional neural network is an iteration of the feed-forward neural network and is predominantly used to process data that has patterns in spatial or grid formats. The most well known application is image recognition where the CNN is very suitable to find edges, colours and transition characteristics. The CNN is also inspired by neuron layer structures, but with some additional types of layers. These layers are a convolutional layer, pooling layer and a fully connected layer. It is not surprising that the input is usually a 2D matrix, or a set of matrices, because spatial relation between the attributes is of high importance. The different layers of the network are explained below.

- **Convolution layer:** The first building block of a CNN is a layer that performs feature extraction from the input data. Typically this layer consists out of two components, a convolution operation and activation function.
  - The convolution operation is a linear transformation where a small matrix passes through the input tensor. This unit is called a kernel which is usually of size 3x3,5x5 or 7x7. The kernel is multiplied element-wise with the input and moves with a given step (stride) through the tensor. The multiplications of the kernel for each stride are summed and assigned to the

feature map. This feature map is a summary of the kernel and the input tensor. Besides changing the size of the kernel, another hyper-parameter is the number of kernels applied. This produces the same number of feature maps as output. Very important is that the weights of a kernel are shared. The kernel is the same for each part of the tensor. This ensures that learned patterns are positional invariant. A feature can be detected in any part of the image. A visual representation of convolution is shown in Figure 5.2.

- After convolution, the outputs are passed through a non-linear activation function. This is inspired by biological neurons that pass through signals with a certain amplitude based on the signal content. In convolutional neural networks the rectified linear unit (ReLU) or a variety thereof is used most often. All non-zero signals are set to zero and all positive signals are passed through as is: $f(x) = max(0, x)$.

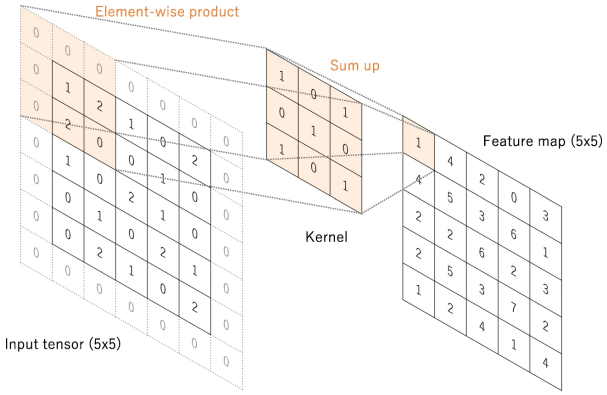


Figure 5.2: Convolution operation visually presented by Yamashita et al.[66]. Note that zero-padding is applied on the input tensor to obtain an equivalent sized feature map. This allows to apply more layers without shrinking the output and losing too much information.

- **Pooling layer**: Similar to the convolutional layer, the pooling layer samples the input matrix (feature map from previous layers) with a specified filter size and stride. This effectively downscales the input to a smaller form factor. This makes the following steps less computationally expensive, but also loses some detail. Pooling is therefor a means to trade off between fidelity and computational cost. Pooling has no learnable parameters like the kernel in a convolutional layer, but relies on the pooling setting. For example averaging all elements for each step (global average pooling), or taking the max value (max pooling). The latter is most often applied, but both methods have their own advantages and disadvantages.
- **Fully connected layer**: The final layer is then a fully connected layer, which is very similar to a generic neural network. However the input of this layer is still in at least a 2 dimensional shape. To pass this through a fully connected layer, a flattening operation is performed. This converts the input tensor to a one dimensional vector. Each input element is passed to the output via a multiplication with a learnable weight. Most of the time there are a couple of fully connected layers in series before the output layer. The output layer is also a fully connected layer, but the number of outputs corresponds to the number of classes. The outcome is then usually a probability of correspondence to each class. The final layer also contains an activation function that is slightly different. For classification this is typically a softmax function that transforms all the outputs to probabilities between 1 and 0 where the total probability sum is always 1.

These different layers together form a convolutional neural network. Using a suitable loss function and back-propagation, the weights in the convolutional and fully connected layers can be learned to produce accurate results. However there are a lot of hyper-parameters that must be evaluated manually, which have a significant influence on the effectiveness of the model. For example the number of layers applied; but also the number of kernels and filters, their size, and stride are of importance. A general overview of an assembled convolutional neural network is shown in Figure 5.3[2].

[2] `https://nl.mathworks.com/discovery/convolutional-neural-network-matlab.html`

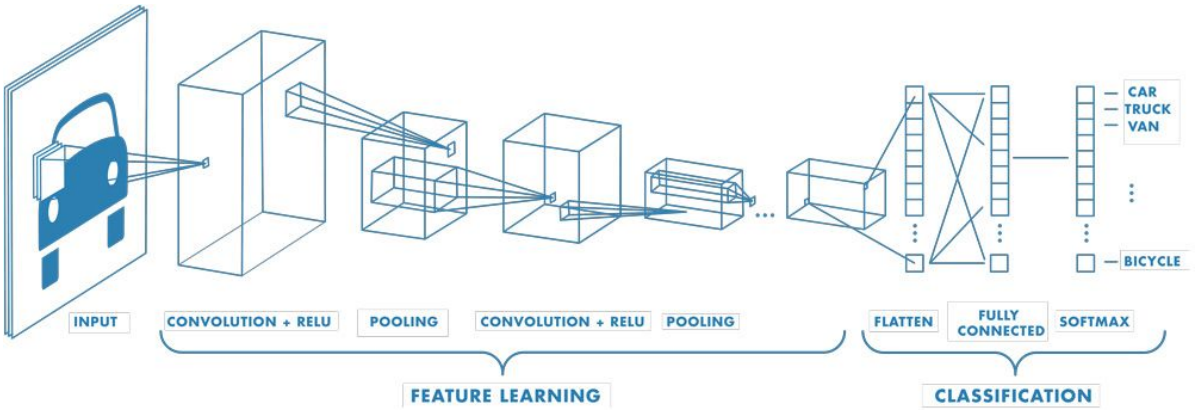


Figure 5.3: Schematic overview of a traditional convolutional neural network used for image classification.

Also in air traffic demand and capacity balancing, convolutional neural networks have proved their usefulness. The first example of this is the research by Xie et al.[64]. In this study, the aim is to predict the complexity of an air traffic sector with a CNN. This is not directly a demand prediction model, but takes the concept one step further and relates it to the workload of an air traffic controller. By modelling the complexity, resources can be applied more effectively. The input data is a set of images generated from air traffic data over a given time interval. The pixels are horizontal locations inside the gridded airspace. Altitudes of aircraft are recorded over the interval and determine the value of the pixels over their flown trajectory. Another channel contains the speeds recorded. The final channel contains the predicted trajectories for the next time window where the 'strength' of the pixel decays when the prediction is farther in the future. If a potential conflict is detected, these pixel values are increased, as conflicts are likely to increase sector complexity. The researchers have recorded the controllers complexity perception, which is used as target variable. The images produced are fed to a classical convolutional neural network that is to classify the complexity score as perceived by the ATCo. The proposed CNN model has a similar layout as shown in Figure 5.3, with 6 convolutional & pooling layers containing a number of kernels between 32 to 128. The model is compared to a variety of machine learning models that are based on hand crafted complexity features obtained from the traffic situation. For example, k-nearest neighbour clustering, support vector machines and logistic linear regression are applied. The results show that the proposed CNN scores best in terms of accuracy (76%), Mean absolute error (0.25) and F1 score (70%). Nonetheless, it is not entirely clear what features the other models are based on. Nevertheless, the error metrics of the CNN prove that it is a sufficiently accurate model to predict air traffic complexity.

An example of a CNN applied to actual flow prediction is the research by Lin et al.[32]. A large block of airspace can be discretised into a 3 dimensional grid. This creates airspace blocks which can be evaluated on aircraft count. These blocks are called a Traffic Flow Matrix (TFM). Because flights generally follow similar patterns due to daily schedules, the flow between these sectors can potentially be modelled with a convolutional neural network, where the input tensor is a traffic flow matrix that is similarly structured as images. However, a classical CNN can only be applied for classification. Yet, the aim is to make predictions on the future state of the TFM. For this, the fully connected layer of the CNN is removed, and a recurrent neural network with LSTM cells is applied instead. A more detailed explanation of the LSTM neural network is given in subsection 5.3.2. This type of network allows to also evaluate sequential data; in this case a sequence of previous traffic flow matrices. The model pre-processes the traffic flow matrix with convolutional and pooling layers. This is then passed to an LSTM network to find temporal relations in the dataset. A simplified schematic overview of the model is shown in Figure 5.4.

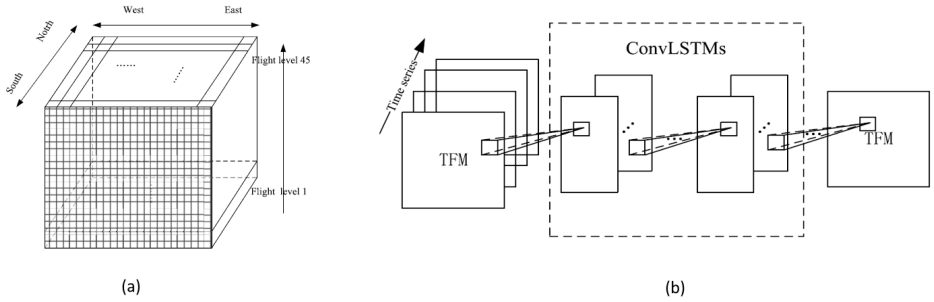


Figure 5.4: The input and proposed model by Lin et al.[32]. (a) The traffic flow matrix which contains discretised airspace sectors with air traffic. (b) The proposed model containing convolutional and pooling layers in conjunction with a LSTM neural network.

Additionally to each Convolutional-LSTM block, a batch normalisation and dropout layer is applied. Batch normalisation is a method to speed the training progress by re-scaling and re-centring the input values for the proceeding layer. Dropout is a technique that can reduce overfitting of the model. It randomly sets some inputs to 0. This effectively introduces noise to the training data, which reduces overfitting of the model. During training a lot of parameters could be specified to design the best performing model. The network configuration in terms of layers, kernels, learning rate and optimiser is to be specified, but also the TFM needs to be found. Finally, a sector spatial resolution is required, and the best suiting time-step must be decided. In the end, a lot of different settings have been tested for both the model and the shape of the input data. The final model is made out of 5 ConvLSTM layers with kernel numbers ranging between 32 and 16. Evaluating a day of traffic in the selected Chinese airspace, it is shown that the model produces less errors during the day then in the morning and evening. This can be explained due to traffic volume increasing and decreasing significantly in this time frame. Another observation is the fact that bigger errors are found in the lower airspace. This is likely due to climbing and descending aircraft, which shows different and more dynamic behaviour than in cruise. The results are compared against three different methods: A regression model, a shallow neural network and a flight plan based model. The proposed ConvLSTM model shows best results with a mean squared error of 80 flights and a variance of 10. The worst performing model is the flight plan based model that has a mean squared error of 120 flights and a variance twice as large. The hybrid convolutional LSTM neural network is therefor a great improvement and may be worth considering for demand prediction applications.

5.2.2. Graph Neural network

A more complex, but very interesting method to predict air traffic flow was introduced by Ma et al.[34]. Their proposal was to model a connected network of sectors with a graph neural network. An airspace system such as the France airspace consists of many sectors that are connected via traffic flows passing through. This resembles a graph, which can be modelled by a so called graph neural network. This type of network is a manually structured network to represent the shape of the actual system. Via propagation over the edges, information can be passed through the network. Ma et al. state that this enables better modelling of spatial-temporal features in hourly, daily and weekly periodic traffic flows. In this section the graph neural network and the specific application to the air traffic flow domain is discussed.

Graphs are real world data structures that can mathematically be reconstructed with three elements: Nodes, edges, and global attributes. A **node** or vertex is a body that connects edges. It can be modelled by an identifier, number of neighbours, and a content value or class. For example when modelling a molecule, a node represents an atom (class) with a number of bonds to other atoms. An **edge** is the connecting element between nodes and can be described by an identifier, a weight and a direction of the connection. Edges enable information sharing between nodes. Lastly the **global attribute** is a summary of the graph. It can contain information such as the number of nodes and longest path. As mentioned before, graphs can be used to describe physical elements in the real world. A molecule is an example of a graph structure, but also images, texts and air traffic networks can be represented in graphs. To model the graph, it may seem straightforward to describe a graph by an adjacency matrix

where each node to node is a row and column and the value in the matrix describes the edges to other nodes. Although matrices can be processed very efficiently by neural networks (see subsection 5.2.1), more nodes cause an exponential increase of the matrix. This is very data inefficient and therefor graphs are described with three different vectors: A vector with the edge attributes, node attributes, and a vector containing adjacency lists. These list contains the actual node to node edges only. The addition of a new node does not cause an exponential increase of the data size. Another advantage of this method is that the Graph Neural Network (GNN) can be evaluated on every element:

- **Global evaluation** can be applied to describe the state of the graph. For example, when the graph represents a molecule, the global evaluation of the graph may predict a molecule property such as colour or smell.
- **Node evaluation** is applied when the state of a node in the graph is to be predicted. For example when a graph represents a network of airports, the nodes may predict the arrival flow of aircraft.
- **Edge evaluation** is applied when the relation between nodes is to be predicted. An example of this can be image recognition, where the image is represented as a graph. The nodes may represent objects in the image. Edge evaluation may then predict the relation between these objects.

The graph neural networks can perform prediction tasks via the process shown in Figure 5.5. First, the graph and its embeddings are transformed, which is shown in Figure 5.5 as a GNN block. This can be done with a a variety of methods. The simplest type of GNN applies a fully connected feedforward neural network on each element. This can be trained via back-propagation similar to any network, and returns a prediction for all elements. Often pooling is applied at the final step as well, making the GNN similar to a convolutional neural network. However, this method does not exploit the properties of the graph structure yet, as the nodes can potentially share information via the edges. This is done via a process called message passing. Message passing is a three step process, where first the embedding of all neighbouring nodes are gathered (messages). Then the gathered messages are merged with an aggregate function (e.g. sum). Lastly, the aggregate is fed to an update function which is most often a neural network. This network updates the graph into a next stage. Applying message passing k-times, the information stored in nodes k-steps away can be fed through the graph. This makes the network capable of learning relations throughout the graph.

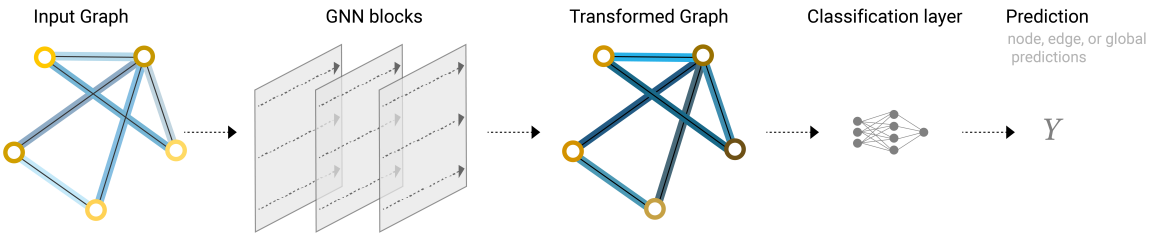


Figure 5.5: Schematic overview of a Graph Neural Network prediction task, created by Sanchez-Lengeling, et al.[48]

Message passing generally is applied to the nodes of the graph, but it may also be applied to edges. A pitfall of message passing is that for large networks information stored in a node farther away then then the number of message passing layers will never be accessed. A solution here is to also include the global attributes in message passing. This works as shown in Figure 5.6, where U_n are global attributes, V_n are node embeddings and E_n are edge embeddings. By pooling information from the global attribute to the update function of nodes and edges, global information of the graph is included in the messages. After updating, the information from the nodes and edges is pooled back into the global attribute.

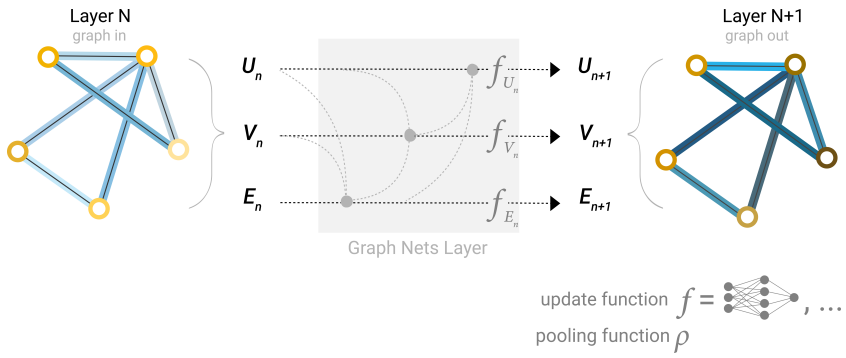


Figure 5.6: Pooling information from the global, node and edge levels in a message passing step may leverage the flow of information throughout the entire graph network. [48]

Graph neural networks are highly modify-able because the update functions and classification layers can be changed for any type of neural network or function. Also, hierarchical graph structures such as multigraphs, or hypergraphs are proposed. In these networks, graphs are nested inside a higher level node to model more complex real world graphs. A challenge for the application of GNNs is to design an accurate graph that resembles the physical object or situation. When looking at the demand forecasting domain, the research by Ma et al. gives a good example. In this research, sector entry flow is modelled with a graph convolutional neural network. The entire model is shown in Figure 5.7. Their model consists out of seven steps that range from processing the raw ADS-B data to making predictions on sector entry flow. First, the training data is processed which is obtained from ADS-B trajectories. The airspace sectors are then overlaid on the trajectories, which results in nodes that describe sector entry flow. The naming format used consists of; first the sector that is entered (S1), followed by the downstream sector (S2) and the upstream sector (S4) of the trajectory. Doing this for all the trajectories gives a large set of nodes.

To generate the edges of the graph, the sequences of nodes are parsed through a Word2vec model. This is a neural network model that originates from natural language processing tasks. It finds associations between words in a provided text. When trained, a word2vec model can return synonyms or associated words on an input word. In this case the words are sector entry nodes from a day of air traffic, so the result will be a vector of most associated nodes. This can be viewed as an edge connecting nodes in a graph. The returned associate vector gives the embedding of the edges.

Together with the nodes, a graph can be constructed. Because the flow changes over time, graphs are created for each time-step where the node embedding represents the temporal flow of the node at that time-step. The predictive model that then follows is tasked to predict the flow at future time-steps. This is done with the proposed attention based spatio-temporal graph convolutional neural network. The input features are composed of three sequences of graphs. First, the recent graph time series, which just includes the last samples before the prediction. The second feature is the daily series, which returns a sample of the same time as the forecast series, but then over the last days to include daily flow patterns. The third feature is the weekly time sequence of graphs, which returns the sequence with the same attributes as the forecast series over the last weeks. So for example, if the model is to forecast on the last Monday of march from 13:00-15:00, then the recent feature contains the series of that day from 11:00-13:00. The daily series contains the series of the previous Sunday, Saturday and Friday during 13:00-15:00. The weekly feature contains the series from the three previous Mondays between 13:00-15:00. Each feature is processed by a few spatiotemporal blocks. This consist of a spatial temporal attention layer (see subsection 5.4.1), and a layer containing a graph neural network and a convolutional neural network. Multiple spatiotemporal blocks are stacked. Finally the three features are fused together with a merge function that also includes learnable weights. Evaluating the predicted nodes for the future time-step gives the expected flow.

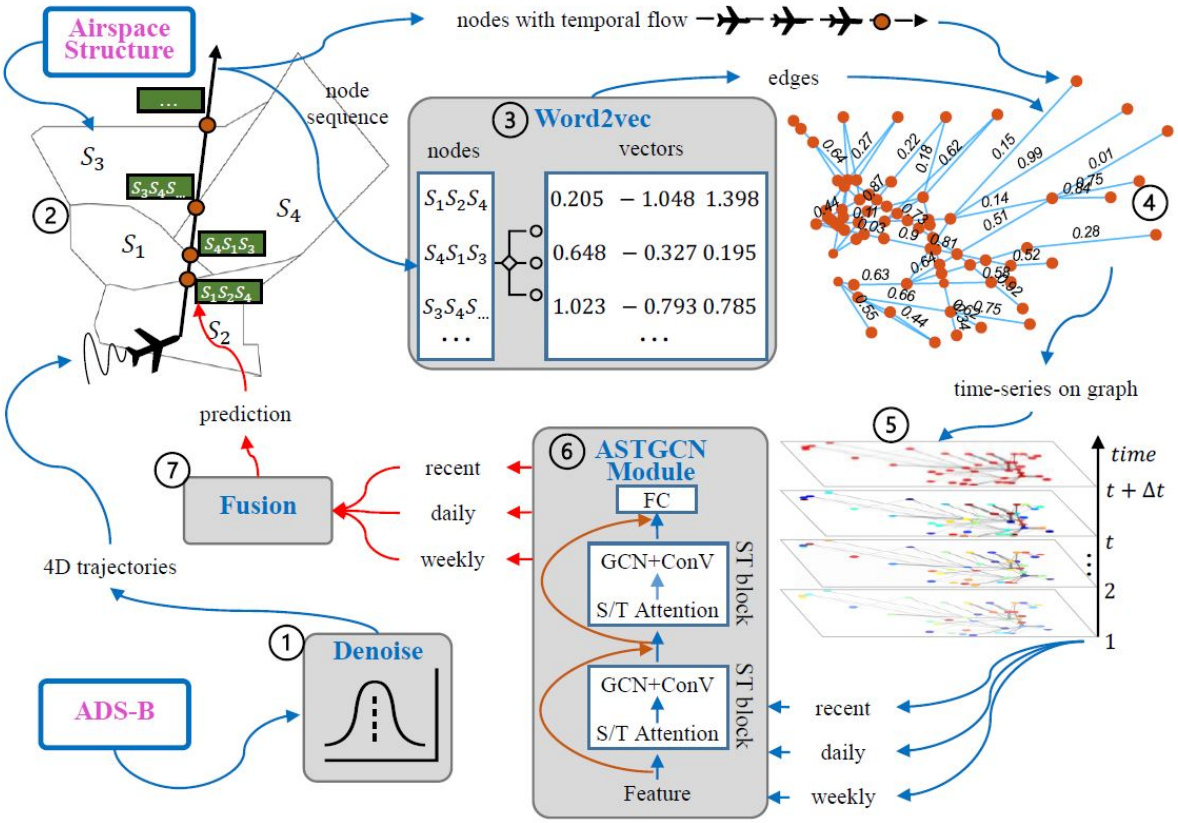


Figure 5.7: Schematic representation of the graph convolutional neural network proposed by Ma et al. [34] to predict sector entry flow.

The results obtained by Ma et al. have already been discussed briefly in subsection 3.2.2. The model was compared against an LSTM neural network for a full day of traffic in French airspace. The Mean Absolute Error (MAE), MSE and RMSE are evaluated. For short look ahead times, the predictions of both models are comparable. However, for longer look ahead times the error of the LSTM based model increase more than the graph neural network. The reason for this according to Ma et al. is that ”the LSTM model learns the correlations in the flow times series on the target node instead of the causal relationship between flows”. The graph neural network has a higher level of awareness of spatial and temporal relations of the entire network and may therefor be able to learn correlations between the node flow and the network flow. This leads to significant improvements at further look ahead time. Hence, the graph neural network is a very promising method to be used in demand forecasting.

Another application of graph neural networks in a very similar domain is the research by Sun et al.[53]. In this study, airport network arrival and departure delay is modelled with a dynamic spatial-temporal graph attention network. Effectively, the model is build up out of a graph neural network and two LSTM layers. Different to the graph convolutional neural network, this network uses an attention function in the message passing process. The model is trained on the European airport network with the top 50 busiest airports with more than a year of data on delay. The model makes delay predictions with look ahead times of up to three hours. The model results are presented in Table 5.1, where the MAE & RMSE are in minutes. Although no direct comparison is made with another method, the authors argue that the graph neural network is a superior method for air traffic network modelling.

Table 5.1: Error metrics for the modelled arrival and departure delay on the European network by Sun et al.[53]

(a) Arrival delay

look-ahead	MAE	RMSE	R^2
30 min	4.60	7.09	0.38
60 min	4.75	7.27	0.35
90 min	4.86	7.41	0.32
120 min	4.93	7.51	0.3
150 min	4.98	7.58	0.29
180 min	5.01	7.63	0.28

(b) Departure delay

look-ahead	MAE	RMSE	R^2
30 min	3.68	5.87	0.35
60 min	3.79	6.01	0.32
90 min	3.89	6.14	0.29
120 min	3.96	6.24	0.27
150 min	4.00	6.29	0.25
180 min	4.04	6.35	0.24

5.3. Machine Learning in Trajectory Prediction

Most of the machine learning models used in trajectory prediction have been discussed already in subsection 4.2.3. A majority of the applied methods are supervised learning algorithms. Regression was used by Hamed et al. [20] and Leege et al. [30] to predict cruise level and time over a waypoint respectively. This is based on observed flight trajectories with a trained regressor. Predicting entire trajectories was proven to be much more complex with regression algorithms. An example of unsupervised learning for trajectory prediction is the Hidden Markov Model, which is introduced in the research of Pan et al. [44], Ayhan et al.[4], and Fernández et al.[17]. This algorithm assumes that the predicted trajectories depend on the observed state. The airspace is segmented into a 3D grid where trajectories pass through the grid. By observing the flight passing through in terms of grid transitions, the model can learn the probabilistic relations between a current and future state of the cells in the grid. The HMM can then generate new transitions representing the future trajectory. However, the trajectory resolution is limited to the size of the grid cells. Lastly, ensemble methods such as gradient boosting machines were applied to short term trajectory prediction by Dek et al.[12]. This type of algorithm creates an ensemble of possible outputs, after which the average is taken as the final result. The learning parameter and different splits in the decision tree determine the model structure during training, such that the cost function is optimised. The results by Dek et al. show great improvements over other methods, but it is non-trivial to apply this model to generative sequences with long look-ahead times.

5.3.1. Clustering

Clustering is an unsupervised classification method that is very frequently applied in Trajectory prediction tasks. Clustering is a method to find and group trajectories of similar characteristics in vertical, spatial or temporal profiles. There are roughly two types of application observed in literature. Trajectory prediction via classification, or clustering to increase the performance of the actual predictor model. The classification approach is not used very often, but a good example is the research by Marcos et al. [36] and Bombelli et al. [6]. In the strategic and pre-tactical phase, when no flight plans are filed yet, this is a very useful way to predict flight routes. Marcos et al. use density based clustering to find similar groups of routes that are flown between city pairs. A choice model then selects the most likely route for a flight between these cities that has no flightplan yet. This can then be used as a dummy in planning processes. An example of the actual trajectories and the clustered routes is shown in Figure 5.8. In this application, clustering is fundamentally used to classify flown routes. When a flight is classified to belong to the cluster, the median trajectory of that cluster is assigned to the flight. In the strategic phase (between 6 to 1 days before operation), this may produce relevant trajectories for planning purposes. In the tactical phase however, the classification method yields trajectories that are too generalised and cannot deviate from the historic set of trajectories. Therefor the uncertainty is too high. For that reason, this is not a feasible method to consider in this research.

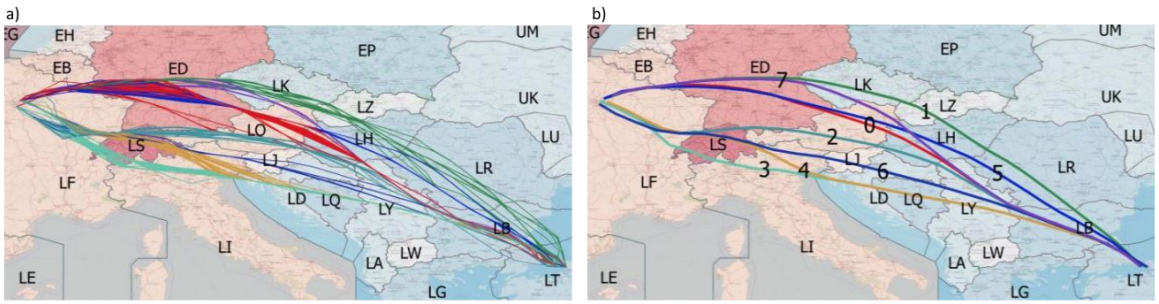


Figure 5.8: Clustering trajectories for route classification by Marcos et al. [36]. a) Actual trajectories coloured by cluster. b) The average trajectories derived from the clusters that can be assigned by the choice model.

In the second approach, other TP models are applied in conjunction with clustering to increase the performance of the actual predictor model. In this hybrid approach, flight trajectories are first clustered to groups after which each group is processed by another machine learning model to produce a trajectory. Many academic research has implemented this approach, amongst which Ayhan & Hamet [4], Fernández et al. [17], Wu et al. [63], and Wang et al. [59]. It is shown that the use of clustered trajectories with specific trained models greatly enhance the predictability. The theory is that different classes of trajectories are exposed to different conditions. For example due to the airspace that the class passes through, weather conditions, or the season/day/time of flight. The different clusters reduce dimensionality and can each be trained with a dedicated model. Usually the model architecture is the same for each class, but training configures the models differently. This allows specific features for each cluster to be captured by the models.

Density Based Clustering

According to Olive and Basora [42], K-means clustering and DBSCAN are one of the most common methodologies to cluster trajectories. These methods are proven to be effective at clustering, but are inherently point based. Trajectories on the other hand are point sequences. To be able to cluster 4D trajectories, a suitable distance function is required. This can for example be the Euclidean distance. Alternative distance metrics can be Dynamic Time Warping, Hausdorff or Fréchet distance. In order to find the similarity, trajectories must be of equal length and therefor require re-sampling. Most recent data driven trajectory prediction studies, such as the TP models developed by Graas[23] & Wang et al.[59], have successfully applied DBSCAN or varieties thereof. Hence it is worth to explore this method in more detail.

DBSCAN is a density based clustering technique that sorts datapoints to groups with high and low density (Ester et al.[14]). The input variables for DBSCAN are Eps & MinPts. Eps is the maximum distance between two datapoints that decides if it belongs to the cluster. MinPts is the minimum number of datapoints required for a cluster. DBSCAN classifies datapoints as either a core point, reachable point, or outlier. Core points have at least MinPts neighbours that lay within the Eps distance. Reachable points can be reached from a core point within the Eps distance, but has less than MinPts neighbouring points. This covers an area with a certain density. Outliers are farther away than Eps and are therefor not included in the cluster. Gariel et al. [19] found that outliers are not uncommon in trajectory datasets and may thus be caught by the DBSCAN algorithm. Especially when using ADS-B data, low quality measurements can be present and must be filtered. DBSCAN will mark such trajectories as outliers, instead of assigning those to a cluster. In contrary to K-means, DBSCAN does not need specifications of the number of clusters. This may be beneficial, as at forehand it is uncertain how many clusters can be defined for the use case of arrivals in the FIR of the Netherlands. On the other hand this can also be a downside, as improper clustering may lead to loss of patterns and useful information alike when similar trajectories are taken apart. Another pitfall of DBSCAN is its applicability to data with high variety of densities. For this, Basora et al. [5] have demonstrated that Hierarchical DBSCAN may be an alternative algorithm.

Hierarchical clustering

Hierarchical clustering is a slightly different process compared to density based clustering. In this bottom-up method, all datapoints are initially an individual cluster. Then the distances between points are calculated and the closest pair is merged to one cluster. This propagates until a single cluster that contains all datapoints remains. A post-processing step is required to select the useful number of clusters for the desired application. Bombelli et al.[6] introduce hierarchical clustering to trajectories. The coarse method is to spatially group trajectories by origin and destination. A finer method however also requires a temporal similarity, because trajectories are 4 dimensional objects and may be flown at different speeds. Therefor the average cruise speed is also taken into account for the fine clustering methodology. Using the Fréchet distance metric, and an automated proces to determine the number of clusters, this research concludes that hierarchical clustering is an effective method to cluster trajectories. This is demonstrated over a region in the US with six control centres and 19 airports.

5.3.2. Long Short-term Memory Network

The trajectory prediction problem is pre-dominantly a sequence generation task. This means that sequential data must be modelled, which is not easily done with convolutional or feed-forward neural networks. The recurrent neural network however is capable to process and output sequential data, hence it is very popular within the machine learning trajectory prediction domain. The basics of the RNN have been explained in subsection 4.2.3. Over the years the RNN is further developed to avoid exploding and vanishing gradient problems. The Long-Short Term Memory Cell (LSTM) cell is one of the advancements made. In Figure 5.9 a graphical comparison is made between a normal RNN block and a LSTM cell. In this figure x_t is the input vector and h_t is the output vector of the current timestamp. Hence the ”vanilla” RNN block receives the output vector of the previous timestep h_{t-1}.

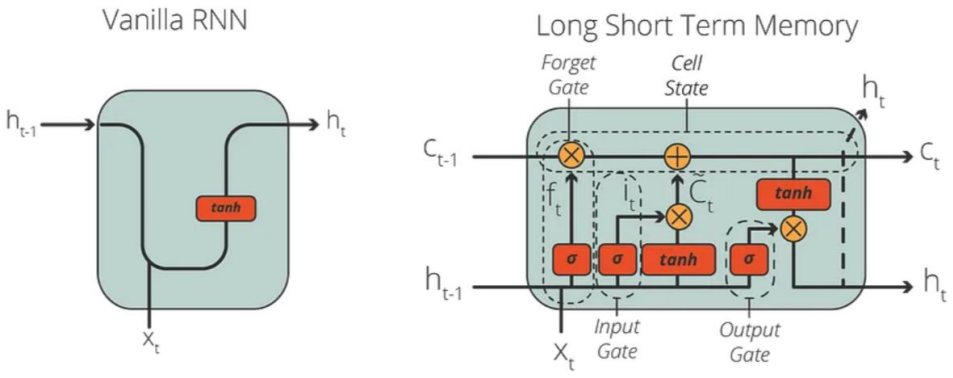


Figure 5.9: Schematic representation of a simple RNN block, and a LSTM block.

The LSTM cell is improved with the introduction of the cell state C_t, which is the horizontal line at the top of the cell. The cell state is a composite signal that can contain information from the previous cell state C_{t-1}, and data from the current cell. There are three gates in that cell that are each activated by a sigmoidal function which passes through either all information (1) or nothing at all (0).

- The first gate is the **forget gate**. This determines to either keep the previous cell state C_{t-1} or to disregard it. This is determined by combining the recurrent signal h_{t-1}, and the input vector x_t, assigning a weight and setting the gate value between 1 and 0 via the sigmoidal function.
- The **input gate** determines whether or not the cell state should be updated. If the cell state is to be updated, then the previous output vector h_{t-1} and the current input vector x_t are passed through a weight and a $tanh$ function and added to the cell state.
- Lastly, the **output gate** filters how much of the cell state is passed into the output. The updated cell state C_t is passed through a tanh function after which it is multiplied by the output gate signal. This is done via the the weighted signal of h_{t-1} and x_t and a sigmoid function, similar to the previous gates.

The LSTM neural network is extensively applied in trajectory prediction. Overkamp [43] used an auto-encoder LSTM neural network to predict trajectories in the upper airspace of Germany while including

air traffic dynamics features. The input data is a discretised trajectory over the last few recorded minutes of the flight. The traffic dynamics evaluated were traffic density, number of heading changes and speed changes amongst others. This shows that LSTM neural networks are versatile tools that can include semi-sequential information as well. With a look ahead time of up to 30 minutes, this model was able to make accurate predictions. However the inclusion of traffic dynamics did not yield higher predictability. Another trajectory modelling approach was for example investigated by Rozendaal [47]. Based on flight plans as an input this model aims to generate new routes that are more true to the actual flown trajectories. The aim of this research was to explore what hyper parameters yield the best results. It was found that a bi-directional LSTM network produced better predictions than an encoder-decoder setup.

Nonetheless, the most relevant study for this research is the generative trajectory prediction model described by Liu & Hansen [33]. Generative models can create a sample of similar structure and characteristics as the samples in the training data. So in terms of trajectory prediction, a model is given some input data X, and output trajectories Y. After training, the model should be able to generate a trajectory from a new input X. This research aims to generate trajectories from flight plans and an initial aircraft state including meteorological data. The model is a hybrid LSTM neural network with some additional data processing components. The first module consists of an LSTM encoder that produces a fixed-size hidden state from the flight plan. The second module is a decoder LSTM network that models the 4D flight state at the next time-step. The decoder has multiple inputs: For training, the input data is the actual trajectory which is processed as a conditional Gaussian mixture from which parameters can be learned. The primary input is the hidden state from the encoder, which contains the flightplan information. Lastly, the additional input is a feature cube from the third module. This module contains a matching algorithm that finds the weather conditions around the aircraft at the current state that is evaluated. This feature cube contains wind, temperature and convective weather information, which is encoded by a convolutional neural network before being fed into the LSTM decoder. Based on the timestamp flight state and features received, the decoder predicts the Gaussian mixtures parameters. During training, the negative log likelihood is used as loss function. However, when using the predicted Gaussian mixtures to sample a new state, this does not yield smooth trajectories yet. For that reason an adaptive Kalman Filter with gating is used. This process generates exponentially increasing trajectories, so beam search is applied to keep the largest sequences only. The Kalman filter and beam search make up the final module. The output of this final module can then be passed into the decoder again to iterate until the final timestamp is reached. The final output sequence with the lowest log likelihood is selected and smoothed with a Rauch-Tung-Striebel Smoother. The lay-out of the entire model is shown in Figure 5.10

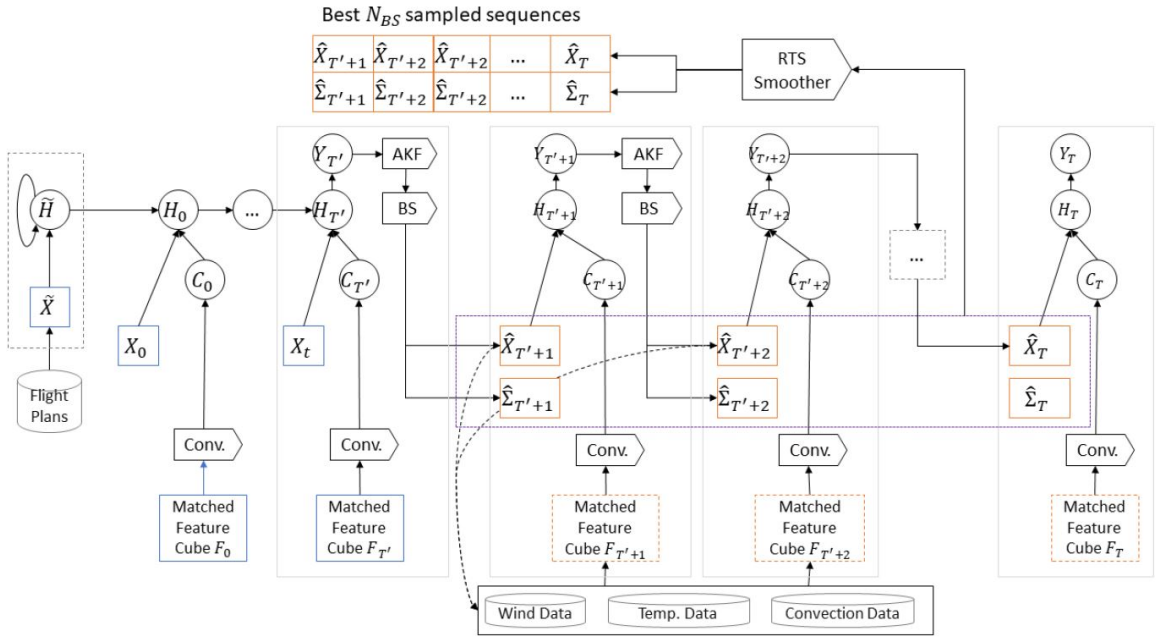


Figure 5.10: Schematic overview of the LSTM based trajectory predictor module proposed by Liu et al.[33].

The experiment performed by Liu et al. scopes down to a single city pair. 1697 trajectories between Houston (IAH) and Boston (BOS) with respective flight plan are used to train and test the model. The mean average horizontal and mean average along track error produced are around 49 nautical miles. The mean average vertical point wise error is around 2800 feet and trajectory wise around 2600 feet. This may seem like fairly large prediction errors, but it must be noted that this is a three hour long flight. Not many similar studies have build generative trajectory prediction models alike. When looking at a visual plot of two predicted trajectories compared to the filed flight plan in Figure 5.11, this prediction is an absolute improvement in terms of routing. When considering the look ahead time and the purpose of the trajectory for demand predictions, the LSTM model of Liu and Hansen can serve as a good baseline.

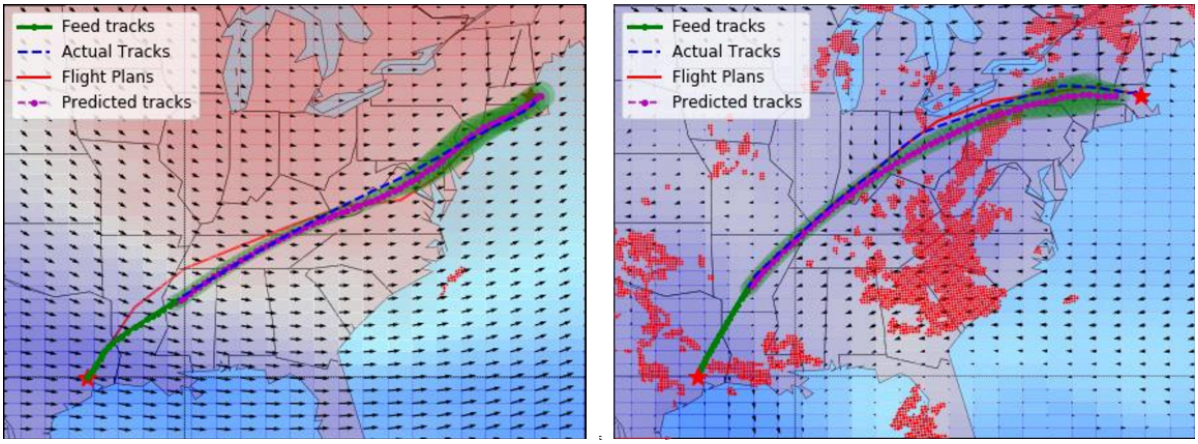


Figure 5.11: Two generative trajectory predictions by Liu et al. [33]. The filed flight plan, actual track and predicted tracks are shown for a situation with strong winds and a situation with convective weather. The blue and red colour scale shows the temperatures.

5.3.3. Generative Adversarial Network

In 2014 Goodfellows et al. [22] proposed the idea of a Generative Adversarial Network (GAN). This is not a fundamental mathematical model like the recurrent or convolutional neural network, but rather a machine learning framework that builds upon basic networks like the encoder-decoder structure.

However, the advancements that GAN networks bring are very interesting and therefor worth discussing in a separate section.

This supervised machine learning network is capable of generating new examples similar to the output data from patterns and regularities in the input data. Goodfellow et al. invented a way to generate a sample that is much like the original dataset. By making the machine learning model compete against the original samples, a high performing generative model can be obtained. This proposal lead to the GAN model that has two primary elements: A generator, and a discriminator. The discriminator is a machine learning model that is first trained to recognise samples in the output domain. For example a discriminator can be trained to recognise an object in an image. When the discriminator is trained properly to correctly classify a true and a false sample, it can be used in the GAN network. The second element is the generator. This is another machine learning model that has the task to generate a sample like those in the output domain. This is done by supplying the generator with a fixed length vector that is of stochastic nature. Sometimes this vector contains a latent variable with noise, sometimes it is just a random vector. The machine learning generator model makes a sample from this vector and feeds this to the discriminator. The discriminator also receives a real sample from the input domain. Now the discriminator classifies which one is true and which is false. If the discriminator made a correct classification, then the generator will be updated. This causes the generator to improve until it fools the discriminator. When the discriminator is fooled and makes a wrong classification, the discriminator is updated. This is a zero-sum game and can possibly be iterated to infinity. The generator eventually becomes very good at creating samples that are plausibly from the domain. A visual representation of the GAN network layout is shown in Figure 5.12[3].

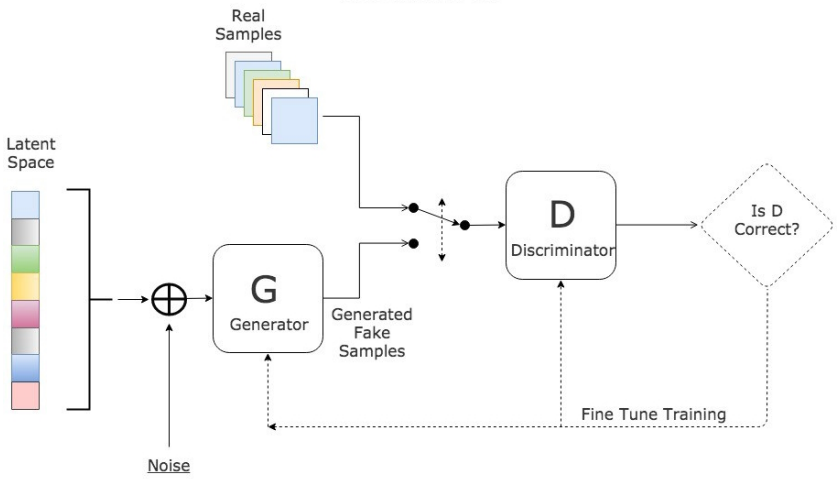


Figure 5.12: Schematic overview of a Generative Adversarial Network.

The GAN network is applied to the trajectory prediction problem by Wu et al.[62]. In this study, the image generation capabilities of GAN models are exploited. First, trajectories are normalised and sampled to a fixed length. This allows the data to be converted to images; where the Red, Green and Blue channels of the image provide enough dimensions to store the data. Original GAN networks are not perfect, because the discriminator can sometimes be instable. When this happens, only a narrow selection of the input is recognised as real. This then propagates to the generator which will scope to produce a subset of the training data only. When training is finished, the generator will only produce a couple of different results. For this reason Wu et al. decided to take a more advanced GAN. The ”Wasserstein GAN Gradient Penalty” model has numerous benefits. It converges faster, fits better to datasets and can be used to a wider range of application architectures. Wu et al. experiment with three different generator/discriminator lay-outs. A one dimensional CNN, a 2 dimensional CNN, and a auto-encoder LSTM neural network. The resultant output of the GAN model is then a newly generated image, which is converted back into an actual trajectory.

[3] https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html

The GAN model is trained to predict trajectories between Chengdu and Beijing (China) on the basis of 2028 flights in the summer of 2019. The 1D convolutional neural network shows to be the most effective at generating trajectories, and is also the fastest predictor. Nonetheless, the error metrics used are not directly comparable to the results by Liu and Hansen [33] for example. The predictions are based on one single flight plan as input and therefor the output is not specifically bound to an actual trajectory or flight. Running multiple simulations returns the average prediction of the model, which is compared to the average of the actual flights. This shows a good improvement over the flight plan. The GAN model is likely to generate improved trajectories, but the error is not directly quantifiable. This method may therefor be an interesting option for trajectory assessment in the strategic domain, but improvements are required to apply this method on the day of operations.

5.4. Alternative Machine Learning Methods

In the previous sections, the most recent and promising machine learning models were discussed that have been applied to demand forecasting and trajectory prediction. Looking back at the general overview of machine learning methods in Figure 5.1, it can be concluded that most of the supervised learning techniques have been applied to the problem. Regression, classification, ensemble methods and deep learning have all been tested to some extent. Most of the research focused on deep learning methods that were a significant improvement over existing demand forecasting or TP models. However, because of the high commercial value, there have been a lot of new developments in machine learning. This results in new models and varieties of existing models. In this section, a few models will be discussed that are relevant to the research.

5.4.1. Transformer Neural Networks

Many of the recent machine learning developments originate from natural language processing tasks. This is a field of research that aims to process and create language elements such as text and speech. Machine learning is a very helpful tool in this field because there is a lot of data available, but language is hard to express through definitive structures and rules. Machine learning was found very capable of explaining the language irregularities. Language processing is often a sequence to sequence task, much like trajectory prediction. For example, translation converts a text in one language to a sequence of text in another language. This requires deep learning methods that can process sequential data. In previous sections one of the most prominent methods was discussed: The recurrent neural network, and more specifically the LSTM neural network, was found to be a good sequence to sequence model (see subsection 5.3.2). Nonetheless, recurrent neural networks such as the LSTM network have a couple of pitfalls that constrain performance of the model. The recurrences in these networks should make sure that historic data influence the predictions, but unfortunately vanishing gradients limit the longer term effect. Also exploding gradients can be present, which block model convergence. Lastly, the biggest constrain of the recurrent neural network type is the computational power required. Each time-step needs to be evaluated separately, which means the computation cannot be parallelised. Parellalisation is specifically important for making efficient use of computer Graphics Processing Unit (GPU) hardware, which is a dedicated powerful computing unit. To overcome these problems, Vaswani et al.[58] proposed the concept of a transformer neural network. This network is able to evaluate a sequence entirely in one evaluation. The internal mechanisms do not suffer from exploding gradients, and the computations can be parallelised. This makes the transformer neural network an interesting model to research.

The transformer neural network invented by Vaswani et al. is structured corresponding Figure 5.13. The left hand side of the model is the encoder structure that takes the input embedding and converts it to a hidden state. The right hand side is the decoder, which learns the corresponding outputs. Based on the learned output, the decoder deciphers the hidden state into an output probability. The primary mechanism in the transformer neural network is the multi-head attention block, which is an ensemble of self attention operations. The entire methodology will be explained in more detail below.

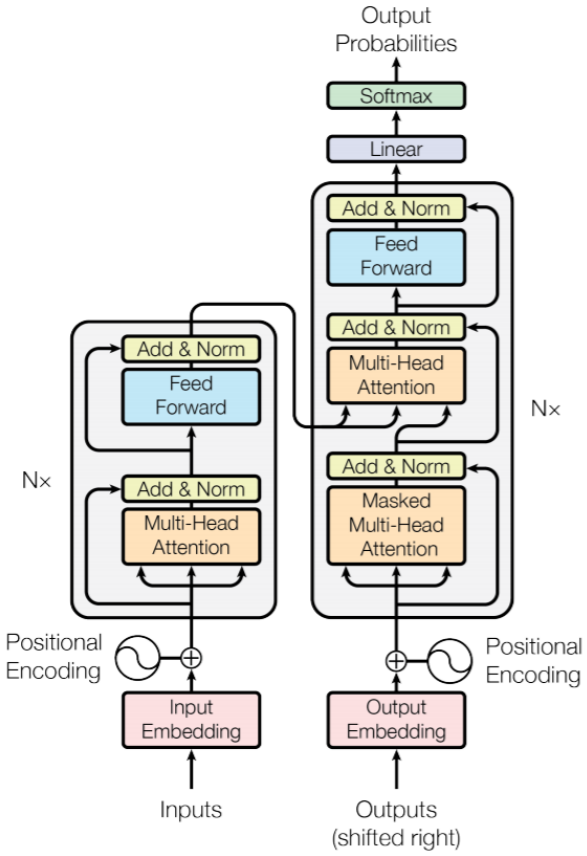


Figure 5.13: Transformer model architecture as proposed by Vaswani et al.[58]

Encoder

The first element of the encoder is the input embedding. This unit takes the input and converts it to a vector that is processable by the model. This conversion relies on an embedding space that spans the whole range of input. Similar items are closely positioned in the embedding space, whereas different items are placed more apart. An example of input embedding is the word2vec model that was explained in subsection 5.2.2.

After the conversion of the input embedding, a positional encoder function adds information to the vector about the position in the input sequence. Vaswani et al. apply sine and cosine functions as in Equation 5.1, but lots of other functions can give similar results. This allows positional information to be taken into account without applying a recurrent neural network. After adding a positional encoding the layers of the encoder begin.

$$\begin{aligned} PE_{(pos,2i)} &= sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{5.1}$$

The signal is then taken into a multi head attention block. The concept of attention is inspired by human visual attention, for example when reading a page of text. While reading, a human pays attention to only a small part of the page to read that content and disregards the other parts. By changing the attention to subsequent words, the whole text can be understood. In deep learning this principle is applied to emphasise the attention of one vector with respect to the other vectors. This is mathematically expressed following the scheme in Figure 5.14(a): First, the input signal is split into three different attributes: Keys (K), Queries (Q) & Values(V). The queries and keys are passed through a MatMul layer which is a matrix multiplication of the queries and keys vectors. This gives a score matrix where higher scores yield more focus, and lower scores yield less focus. Then the score matrix is scaled with the square-root of the dimension of the matrix. This reduces exploding values when evaluating lots of

different vectors. Optionally the score matrix can then be masked, but this only happens in the decoder of the transformer. Finally, a SoftMax function is applied, which gives a matrix of probabilities where each row or column adds up to 1. In the final step, the original signal (values) is then multiplied with the attention weight matrix. The higher attention scores will thus keep the signal of the vector high, whereas the lower attention scores will reduce the respective signal.The mathematical expression of attention is given in Equation 5.2. Because the Queries, Keys and Values all originate from the same input vectors, the operation is called self-attention.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{5.2}$$

In multi-head attention, multiple attention evaluations are applied in parallel with a linear feed forward layer before each attention head. Afterwards the produced vectors are concatenated and a linear layer is applied. This is shown in Figure 5.14(b). The linear layer introduces learnable weights to be able to train the signal. The final concatenation and linear layer combine the result of all attention heads into a single output vector with the same dimensions before the multi-head attention block. Because each head has learned differently, the final vector can have a lot of representative power. Vaswani et al. implement 8 attention heads, but this can be a different design parameter.

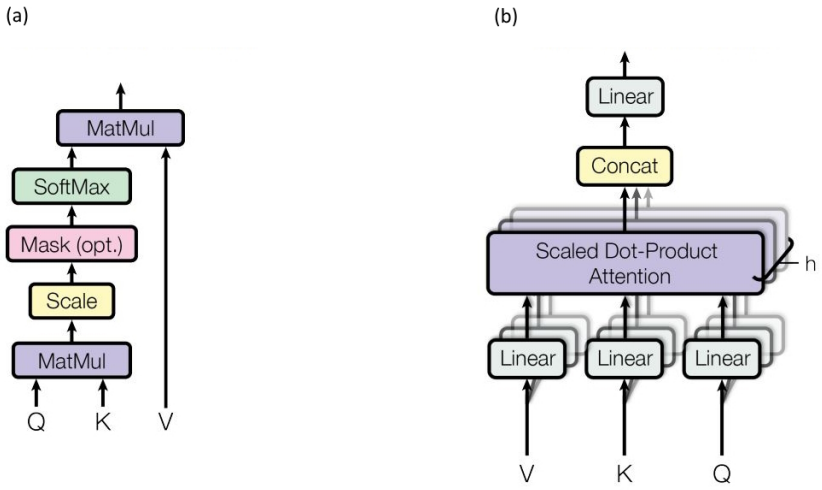


Figure 5.14: The attention mechanism as presented by Vaswani et al.[58]. (a) Scaled Dot-Product Attention. (b) Multi-Head Attention.

After the multi head attention block, a residual flow of the input is added and the result is normalised. The following layer is then a position-wise feed-forward neural network. It contains two linear transformations with learnable weights, and a ReLu activation function in between. The residual taken before the feed-forward layer is then once again added to the output of the feed-forward layer, which is then normalised. The number of encoder layers can be selected as a design parameter. Vaswani et al. apply six layers for their language translation task.

Decoder

The encoder and decoder are very similar in structure with multi head attention blocks and feed forward layers. However, the decoder has small but fundamental differences. The decoder is an auto-regressive function, meaning the signal is processed one element at the time. The output labels corresponding to the input are shifted by one position first. This means that, given the input at position i, only the previous outputs $(i-1)$ are made accessible to the decoder. Otherwise a 1-to-1 mapping may be established by the model, which does not yield any predictive power. The signal is taken through the embedding space which results in a vector format. The positional encoding is added to the vector to give positional context information.

Similar to the encoder, the signal is then taken into a multi-head attention block which applies self attention. However this time, the optional masking element as shown in Figure 5.14 is in place. Masking

is applied before the SoftMax function, and sets values that should not yet be available to the decoder to negative infinity. Together with the position offset, the masking makes sure that the model will not be attending to a signal that is yet to come. This block also contains multiple attention heads which are then concatenated and fed through a linear layer. The residual signal is added once again and the result is normalised.

The decoder then has another multi-headed attention block, but for this block, the queries and keys are taken from the hidden state output of the decoder. The values are taken from the decoder signal. This allows the model to determine which part of the encoder input the focus should be placed on. All the encoder input is available, so the decoder can attend over any position. The result of this multi headed attention block is added to the residual decoder flow and normalised.

The following layer is equivalent to the encoder, with a position-wise feed-forward neural network, and a layer that adds and normalises the residual signal. The final layers of the decoder are a linear feed-forward layer that acts as a classifier. The input is mapped to N output classes. Lastly a softmax function is applied once again that converts the classification into a probability score between 0 and 1. The index of the highest probability can then be taken as the predicted value.

Results

As mentioned, Vaswani et al.[58] build the transformer neural network for natural language processing purposes. The experiment was therefor to make translations of sentences. The model was trained on the WMT 2014 English-German and English-French datasets which has millions of sentences in both languages. An Adam optimiser was applied with a variable learning rate. Also during training, dropout and label smoothing were used to tweak model performance and reduce over-fitting. The results show a superior score for only a quarter of the training data compared to other state of the art translation models. The main idea of transformer neural networks was that computations can be parallelised, and therefor show faster convergence. This is an advantage of the attention mechanism compared to recurrent signal processing in RNN networks. This hypothesis indeed proved to be the true. Comparing the results of 8 different models, the performance scores achieved of the transformer models are often better with significantly lower training costs.

The transformer neural network is included in many commercial applications such as Bidirectional Encoder Representation from Transformers (BERT) and Generative Pre-trained Transformer (GPT). BERT can perform lots of specific tasks in natural language processing. For example text classification or language inference. GPT, is a similar tool for language processing tasks. It was recently made available for public use as a text generation machine, that produced outstanding results [4]. Applications outside the natural language processing domain are also found. In biology, lots of sequential data exists. DNA, RNA and protein structures are just a few examples of this. Zhang et al.[68] survey which biological modelling advancements have been conceived by transformer neural networks. Especially the BERT model was modified to many different tasks. For example a significant improvement was made on identifying causal factors on acute liver failure. Another BERT adaptation, named GeneBERT, showed good improvements on tasks such as promoter and transcription factor binding sites classification, which is an important field in DNA and protein studies. Lastly, transformers have also found their way to transportation engineering. An example of this is the study by Wen et al.[61], where a convolutional neural network is combined with a transformer to predict road traffic flow. The model once again shows state of the art performance with a reduction of computational cost compared to recurrent neural networks.

5.4.2. R-Transformer

The groundbreaking improvements that were conceived with transformers led to a variety of different implementations. The heart of the transformer, the (self) attention mechanism, was applied in for example the graph neural networks by Ma et al.[34] & Sun et al.[53]. This was discussed in subsection 5.2.2. But also entirely new model structures have been created based on the transformer neural network. A good example of this is the recurrent neural network enhanced transformer, also referred to as R-Transformer, proposed by Wang et al.[60]. They argue that transformers are very effective at capturing long term dependencies in sequence-to-sequence prediction, but the transformer lacks components

[4] https://openai.com/blog/chatgpt

that can capture very local dependencies. Also, transformer models tend to be very design intensive, meaning it is not straight forward to find the right settings for a model converging on the optimum. The R-Transformer was designed to overcome this problem by combining the strength of recurrent networks to model local dependencies, and the strengths of the transformer in computation speed and longer term dependencies. The proposed machine learning model is composed out of three different layers: A local recurrent neural network layer, a multi-head attention layer, and a feed-forward layer. An illustration of a single R-Transformer layer is presented in Figure 5.15.

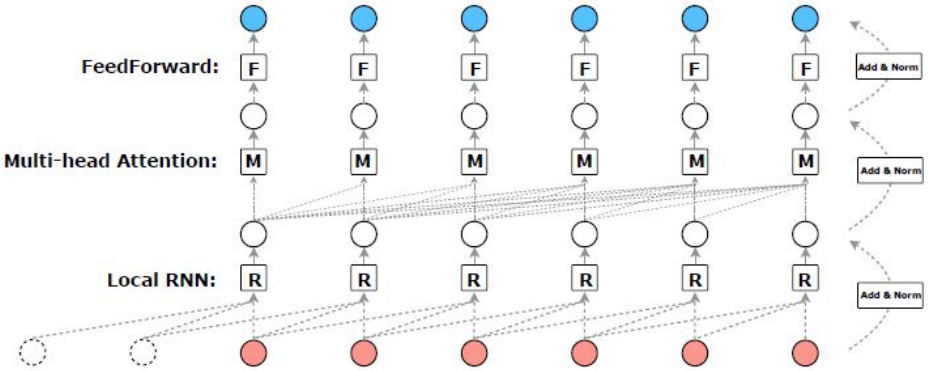


Figure 5.15: Schematic overview of a single R-transformer layer by Wang et al.[60]

To better represent local dependencies of language sequences, the R-Transformer was designed with a local RNN as first layer. The full input sequence is split into short batches that are sequentially evaluated by an RNN. This makes the operation analogous to a convolution operation. The local RNN can be designed with standard recurrent cells, but other varieties such as GRU & LSTM cells are also possible. The subsequent layer is then a multi-head attention layer which is equivalent to the original transformer by Vaswani et al.[58]. The final layer is a position-wise fully connected feed-forward layer which is a non-linear transformation of the multi-head attention output. Lastly a normalisation layer is applied to make the final prediction bounded. To show the effectiveness of the model, a three layers deep R-Transformer is compared to other similar machine learning models. Amongst which a transformer, temporal-convolutional network, and LSTM neural network. The comparison is made on three different tasks: Pixel-by-pixel sequence classification, polyphonic music modelling, and a language modelling task. The R-Transformer is shown to be the best performing or runner-up model in each task. However it is unclear what computational strain this model induces, which is one of the benefits of the original transformer over recurrence based models. Lastly, the tasks presented are not generative sequence-to-sequence problems. If the R-Transformer is to be applied on pre-departure trajectory prediction, an adaptation is required to make this work.

5.4.3. Token Mixing

Another proposal in machine learning to improve over the success of the transformer is token mixing. As explained before, Transformers rely on the attention mechanism to find relational context between elements in the sequence. Because of the scaled dot product, the attention mechanism complexity increases quadratically with the length of the input sequence. This means that the computational cost increases quadratically as well. To reduce the cost, lots of modifications and tweaks have been made to the transformer. Other developers and researchers began exploring different means to capture relational context within the sequence. This is where token mixing models are introduced. Instead of having an attention layer, these models introduce a layer that converts or changes the input sequence to another format, without losing important sequential information. For example, Lee-Thorp et al.[29] introduced a Fast Fourier Transformation layer to replace the attention layer. This converts the input sequence vectors to a vector representation in the frequency domain. Because part of each sequence token will be represented in a frequency vector, tokens are mixed. This allows relational context to be learned in the subsequent feed-forward layer. Because the Fourier transform is reversible, all information is maintained, only in a different domain. An example of the encoder structure is shown in Figure 5.16(a).

Another token mixing proposal was brought forward by Tolstikhin et al.[56], as an alternative to transformers and convolutional neural networks in computer vision applications. This machine learning model, called MLPmixer, is relying on token mixing through a static multilayer perceptron. A multilayer perceptron is a feed-forward neural network with at least one hidden layer. The proposed model takes the sequence of pixels as input and divides it into N patches. The patches are then transposed and passed through a multilayer perceptron. This layer consists of two fully connected feed-forward neural networks with a non-linear GeLU activation function in between. The output is then transposed back. This has effectively mixed the tokens, without loosing any information. The original input patches are added as a residual flow and the result is then normalised again before another mixing layer is added. An illustration of the MLPmixer is shown in Figure 5.16(b).

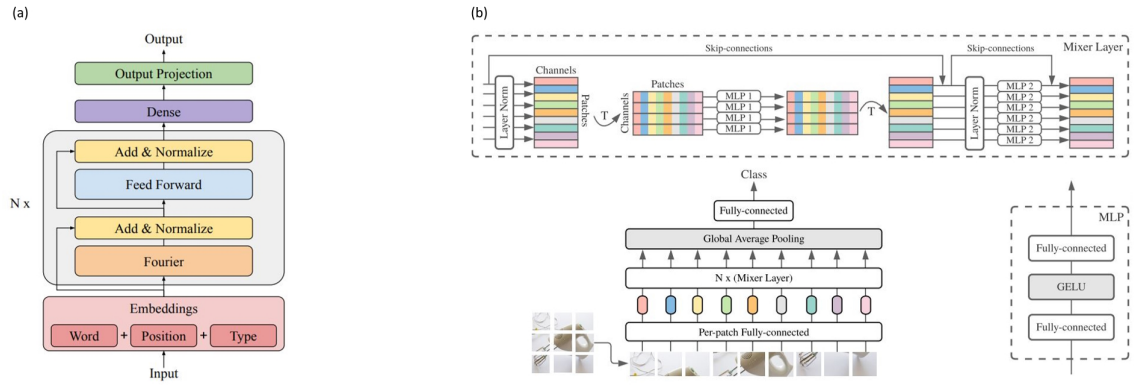


Figure 5.16: Token mixing models have been proposed to reduce computational costs of current operational machine learning models such as transformers. (a) Lee-Thorp et al.[29] introduced the Fast Fourier Transform as an effective replacement for attention. (b) Tolstikhin et al.[56] used a transpose operation in combination with multilayer perceptrons as a token mixing layer.

The results for both the Fourier based model and MLPmixer are close to current state of the art methods. Depending on the scenario in which they are applied, token mixing methods may yield equally sufficient results as the best performing transformer neural network. However, the computational cost or time can be decreased significantly with token mixing. Lee-Thorp et al.[29] show that their Fourier transformation network is almost twice as fast as the evaluated transformer models. When selecting a model for sequence modelling applications, the costs should not be neglected. Token mixing models can be a great alternative in that regard.

6

Data Collection and Preparation

In order to improve demand forecasting of a specific air traffic control sector, data is an essential element. Given that this research is to improve the demand predictions with machine learning methodologies, the data provisions are at the core of this research. In chapter 3, the current state of the art in demand forecasting was discussed. Various sources have shown the predictive accuracy, and what elements contribute to the observed error. Based on these findings it became clear that departure time uncertainty and trajectory prediction errors are at the root of the problem. The selected approach in this research is to apply a novel machine learning model to predict trajectories. chapter 4 has shown different methods of trajectory predictions that have been tested. Clearly, model based TP is unable to include deviations on the longer look ahead times that are required for demand forecasting. Although this may be solved with improvements of intent information, this is not yet possible. Alternatively, data driven trajectory prediction may be able to capture these differences through pattern recognition and statistical evidence. For this reason it is decided to build a trajectory predictor model based on machine learning. The selected model will be a transformer neural network. However, before modelling can kick-off, it is important to investigate what data is required, what data is available, and how to process it. For this research two primary sources of data are available. First in section 6.1, the available information as model input is explained. This is data from the Eurocontrol Network Manager via the Business-to-Business (B2B) connection that LVNL has. In order to train and validate the model, the actual trajectories must be available as well. For this, ADS-B data is used that is provided by the OpenSky network. This data is explained in section 6.2. These two data sources are at the core of the model. For improvements however, weather information will be very relevant. This data is retrieved from the European Centre for Medium-Range Weather Forecasts.

6.1. B2B Data

The Eurocontrol Network Manager is an organisation that is centrally located regarding European air traffic information. Because they are monitoring the traffic flow across the entire ECAC area, they receive a lot of information from ANSPs, airlines, airports, ground handlers and other stakeholders. Because the European Union is developing towards a Single European Sky, most of this information is shared by the network manager. For securely sharing the data with the right stakeholders, different networks and architectures, such as SWIM and business to business connections have been build. LVNL, the Dutch ANSP, is subscribed to the B2B connection with the network manager. This connects the organisation with lots of different information flows. The information flows are categorised into different services which are explained below.

- **Flight services:** Specifically focused on actual flights towards and inside the ECAC area. This service provides data sharing capabilities on flight planning and management between airspace users and the ANSPs. It allows users to create and file a flightplan, which is then validated and monitored with the associated air traffic control sectors. Furthermore, flight services include departure and arrival planning tools which is relevant for (CDM) airports. Air Traffic Flow and Capacity Management (ATFCM) slot information is another important element in the flight services

data suite. This allows the network manager to regulate flights for demand and capacity balancing. Lastly, flight services also contain information about the progressing of airborne flights. This can be position reports, but also status updates or system activation messages.

- **Airspace services:** Via this service, NM is providing a means to access up-to-date information on airspace. This data is retrieved from the Aeronautical Information Publication and NOTAM provisions in the European airspace. Also availability of airspace is included which should allow Flexible Use of Airspace implementation.
- **Flow services:** The primary function of the network manager is to manage the air traffic flow through the European network. With this service the network manager updates stakeholders on the regulations that are enforced. As this is a collaborative effort, there are several management tools in place to manage ATFCM scenarios and measures, or simulate network impact. This service also shares information such as traffic count at aerodromes and airspaces or delay situation information.
- **General information services:** The final service contains general information about the B2B service. This includes general ATFM messages about the global network operation. Also release notes and technical documents on the B2B services are available here.

From the above-mentioned services, the flight services and airspace services are the most relevant information sources for building a machine learning trajectory predictor. The flight services that are available to this research consists out of flight messages containing information about flights. This data is received in XML format via a secured web subscription service. For this research, given the look-ahead time horizon of 3 to 5 hours in demand and capacity balancing, the most important messages are Flight Plan messages and Position Reports. Flight plan messages can be First System Activation, or ATC Proposed Flight plan messages. These messages contain the scheduled flightplan, which is a sequence of waypoints that have a latitude, longitude, flightlevel and estimated overhead time. Besides this information these messages also contain general flight information such as callsign, departure and arrival airport, aircraft type & registration, and timing information. This includes an estimated off-block time, arrival time and taxi time. The flight plan however is not only included as a flightplan format, but the message also includes an airspace crossing plan. Based on the flightplan, the estimated times for airspace crossing are calculated. This allows the network manager and ANSPs to directly obtain demand estimates from the flight plans.

For flights that are already airborne, the B2B data contains regular updates via a Correlated Position Report (CPR) message. This is very similar to the flight plan messages, with exactly the same data-fields. However additionally the message contains a position observed by a connected ATC surveillance data processing system. The actual position is a geographic 4D position in latitude, longitude, altitude and time. Alternatively to the Correlated Position Report, the aircraft operator can also provide information when airborne. Long haul flights that are outside of the Network Manager Operation Centre can send its current geographical position, or an estimated arrival time to the network manager via the Aircraft Communications Addressing and Reporting System (ACARS). This can be included in a message as well, which is called an aircraft operator position report. These messages are enriched with data-fields similar to flightplan or CPR messages. In the future, the aircraft operator position reports may become a more frequent type of message, as more and more aircraft are equipped with ADS-C. This air-ground data-link allows the aircraft to share its intended flight profile as a 4D trajectory with an ANSP, airline operation centre or the network manager.

6.2. ADS-B Data

Automatic Dependent Surveillance-Broadcast (ADS-B) is a surveillance system that is periodically transmitting aircraft position and states parameters. The signal can be received with relatively simple receivers on the ground, in other aircraft, or even by satellites. The technology does not rely on interrogation signals by radar systems, as the onboard squitter equipment is broadcasting automatically. In the future, ADS-B systems may be able to replace secondary surveillance radar, and the technology is mandated by the FAA and European Union Aviation Safety Agency (EASA)[10]. This has caused very high levels of equipage on aircraft nowadays. Because the broadcasting signals are not encrypted, any receiver of the 1090MHz frequency can receive and decode the ADS-B messages. There are a variety of parties that have implemented a network of receivers covering large parts of

the world. An example of such a network is the OpenSky Network. This organisation has a historic database of ADS-B messages to which researchers can apply for access. This thesis thankfully makes use of the data provided by the OpenSky Network.

ADS-B messages are 112 bits long and have five main parts: The down-link format, transponder capabilities, aircraft unique ICAO address, the extended squitter message, type code, and Parity/Interrogator ID [52]. For this research the ICAO address is important to collect information of entire flights. The type code is used to identify the contents of the broadcast, and the extended squitter message contains the parameters itself. Table 6.1 shows what parameters are transmitted in the ADS-B messages. Note that the table is only a summary of the primary contents. For efficient and reliable transmission, some messages are encoded and/or split over multiple messages. In this case some bits are reserved to include decoding keys. These details are left out of scope, as the data received from the OpenSky Network is already decoded.

Table 6.1: Summary of the most relevant data parameters included in the \ac{ADS-B} messages

Type Code	Data Frame Content	Variables	Unit
1–4	Aircraft identification	Call-sign Wake Turbulence Category	- -
5–8	Surface position (on ground only)	Latitude Longitude Ground Speed Track Unix timestamp	deg deg kts deg s
9–18	Airborne position (Baro Altitude)	Latitude Longitude Baro altitude Unix timestamp	deg deg ft s
19	Airborne velocities	Vertical rate Ground speed (North/East) Airspeed (North/East)	ft/min kts kts
20–22	Airborne position (GNSS Height)	Latitude Longitude GNSS altitude Unix timestamp	deg deg ft s
31	Aircraft operation status	Operational capabilities Data integrity & accuracy Track	- - deg

The ADS-B broadcaster sends this information according to a specific scheme, depending on the state of the aircraft and transponder setting. Combining multiple broadcasts with different information can reproduce the actual state of the aircraft. The OpenSky Network obtains the raw messages from the connected receivers and decodes this into complete messages. The data is enriched with for example the aircraft type and registration, which can be obtained from an ICAO id aircraft database. Call-signs and flown tracks can be used to find origin and destination airports. The Traffic python toolkit developed by Olive [41] is used to access the ADS-B data. As explained in Table 6.1, the messages contain the actual 4D positions of aircraft: Latitude, longitude, altitude and timestamp. This data will be the reference for the machine learning model to train on.

6.3. Weather Data

Flight trajectories are highly influential to weather conditions. As aircraft physics and sensors are expressed in a reference frame relative to the air-mass, changing wind conditions will directly change the ground speeds. In turn, this results in different trajectories. Furthermore, convective weather such as thunderstorms are usually avoided, as they may interrogate safe flight execution. Lastly, also temperature plays a role in the performance of the aircraft, particularly the cruise speed and climb performance. All these elements may contribute to deviations observed between the planned trajectory and actual tra-

jectory. It is therefor decided to explore the possibilities of predicting trajectories given a set of weather conditions. Having studied the results of Liu et al.[33], their machine learning model is capable of including weather conditions ahead of the trajectory via a grid matching algorithm. This grid contains the southerly and westerly wind components, air temperature, and convective weather parameters for 13 altitude levels and a spatial resolution of 20x20 nautical miles. The weather forecast and current conditions are updated every four hours, with predictions for every hourly window. This data comes from the North American Mesoscale Forecast system and the National Convective Weather Forecast system. Although implemented successfully, these datasets are not available to the European area of interest in this thesis. A European alternative to the wind and temperature data can be obtained from the ECMWF ERA5 database. This is a publicly available dataset that has hourly estimates of meteorological variables. The spatial resolution of the model is 30km and the pressure altitude resolution contains 37 levels ranging from 1000hPa to 1hPa. Because aircraft usually do not fly at such high altitudes, the levels up to approximately FL480 can be taken. The standard atmosphere pressure formula is given in Equation 6.1, and temperature equation in the troposphere is given in Equation 6.2 (Anderson[2]). Where P is a pressure at a given altitude, g_0 the earths gravitational acceleration, T temperature, $\frac{dT}{dh}$ temperature lapse rate with altitude, h the altitude itself, and R is the gas constant.

$$\frac{P_{alt}}{P_0} = (\frac{T_{alt}}{T_0})^{\frac{-g_0}{\frac{dT}{dh}R}} \tag{6.1}$$

$$T_{alt} = T_0 + \frac{dT}{dh}(h_0 - h_{alt}) \tag{6.2}$$

With these equations, knowing the temperature lapse rate, ground level temperature and gas constant in the troposphere, it can be determined that 125hPa is a suitable maximum level. This leaves 26 discrete levels with meteorological conditions that can be used. The ERA5 database has a spatial resolution of 30x30km and contains temperature, vertical air movement, northerly wind & easterly wind components. In this respect a similar or even better resolution can be obtained. However there is no convective weather dataset publicly available for the European situation. In case there is an opportunity to implement this data, a new grid may have to be established to merge the wind, temperature and convection attributes. The available features of the ERA5 weather dataset are given in Table 6.2

Table 6.2: The weather data features provided by the ERA5 dataset of the ECMWF. The grid resolution is 30x30km, with 26 pressure levels. The data is obtained from hourly observations.

Variable	Unit	Description
Air Temperature	K	Average temperature of the local atmosphere
V wind component	m/s	Northerly wind component (North positive)
U wind component	m/s	Easterly wind component (East positive)
Vertical air velocity	Pa/s	Vertical velocity of the air mass in the grid. In Pascals (pressure altitude) per second. (Downward motion is positive)

6.4. Filtering & Pre-processing

As with any dataset there are always impurities, missing entries, or outliers. Before model developments start, it is important to analyse the data and to create a development strategy. Data can only be analysed when it is available in the right format. Furthermore it is important to filter out the samples that are erroneous or unsuitable. Different levels of pre-processing and filtering are required for each dataset. For example, the B2B data requires a lot of preparation. Once extracted, the data is already of high quality, so only a few filtering operations are required. For ADS-B data, this is the other way around, with an emphasis on filtering. Nonetheless, it is important to first determine what data is required exactly. This research aims at generating trajectories at 3 hours before arrival in the Dutch FIR. For most flights this means that the aircraft is still on ground, hence the entire trajectory is to be generated. The selected model will be a transformer neural network type, which is capable of predicting new sequences from an input sequence, based on learned output sequences. The model will be build in python with the PyTorch library [1], which requires input in a [sequence, batch, feature] format. The

[1] https://pytorch.org/docs/stable/generated/torch.nn.Transformer

batch contains a set of flights, the sequence are timesteps and the features can be any type of data that influences the actual trajectory. In essence, this translates into a flightplan as input time sequence, where the features are latitude, longitude and altitude. The desired output sequence is a trajectory with the same features, albeit with a different length and temporal resolution. The sequence which the model should train on must therefor be the corresponding trajectory from the ADS-B data. This means that the primary input flightplan from the B2B data should be matched with the actual ADS-B trajectory. When the messages are matched with the actual recorded flights from the B2B data, the actual arrival time at the FIR boundary will also be known. With this information the latest message three hours before arrival can be selected.

6.4.1. Pre-processing for analysis

B2B data preparation

In this phase of the research, a dataset of B2B messages from the Eurocontrol Network Manager to LVNL during may 2021 is available. Because of the COVID-19 pandemic, traffic numbers are lower than in the years before, with around 400 flights each day. It would be preferential to have a dataset that covers a more representative period, but this data is not available as of yet. The messages are provided in an XML format, which must first be parsed into a python processable format. During the entire project, the pandas library [2] is used with a data-frame structure that contains the data. The dataframe is then filtered for arriving flights only. The flight plan does not contain the locations of waypoints, which are looked up with the Eurocontrol navigation and airspace database. The latitude and longitudes are converted to degrees, altitudes are rewritten to flightlevel, and timestamps are specified in UTC date-time. Besides having the actual scheduled time at the waypoints, a flight duration time is calculated as well. This is calculated based on subtracting the scheduled take-off time from the estimated overhead times. Additional information in the B2B messages such as timings, aircraft parameters and airspace crossings are also saved. Part of this information is used to match the corresponding ADS-B trajectory to the message. This is done with the flight callsign and the day of the flight. Other information contained in the B2B messages is saved, but not yet prepared. There may be valuable information that can improve the model, but this will be explored once more complex varieties of the model are build. The final step is to find the FIR arrival time from the flight plan. This is done by re-sampling the flightplan at a 10 second interval via interpolation between the waypoints. The resultant is then checked as to which part is inside the FIR airspace. The final data-point before entry is selected as the FIR entry time. This is the most important element for the demand predictions.

ADS-B data preparation

The ADS-B data is retrieved from OpenSky with help of the traffic package developed by Olive [41]. This is a python based toolkit that has numerous data processing and filtering capabilities, which make it a straightforward operation. A query is made that retrieves all flights recorded to arrive at Amsterdam Airport Schiphol for each day in the B2B messages dataset. As Sun[51] explains, almost half of all the ADS-B messages are corrupted. Although the retrieved data from OpenSky is already pre-processed, there may still be errors present. A filtering operation is applied that can effectively remove most outliers. Missing datapoints are also fairly common, especially in areas where coverage is harder to accomplish. For example the Atlantic ocean, or less populated continental areas. The trajectories are therefor smoothed and re-sampled every 10 seconds, which interpolates between upstream and downstream datapoints. In this research the FIR entry time is the primary element for demand predictions. Therefor the trajectories are checked as to which part is inside the FIR. This part is removed and the last data-point of the remaining trajectory will be taken as the FIR entry point. Finally the actual trajectories are matched with the B2B messages. With the flight callsign, date, and registration, the flights can be matched correctly. The actual FIR entry time can be added to the B2B messages. This allows to select the latest message three hours before arrival, as this message will be the main model input.

6.4.2. Pre-processing for neural network modelling

For neural networks such as the transformer to be effective, the input data quality is of utmost importance. In principle, all neural networks learn to recognise statistical patterns within the training dataset and project this on newly derived samples. Consequently, a well-known statement associated with neu-

[2] `https://pandas.pydata.org/`

ral networks is that garbage input will yield garbage output. For this reason a lot of effort must be put on selecting the right variables and providing these variables in a proper format. The selected transformer neural network will be build in the python PyTorch environment. The input for a transformer neural network, or any other type of sequence modelling network is of the structure [sequence, batch, feature]. Where the batch contains different flights, the sequence contains the timestamps and the features are to be defined. In previous sections it was already explained that the primary features are 4D position sequences. First a geographical coordinate system transformation is required to effectively model lateral positioning. Other required feature engineering practices are one-hot-encoding for non-sequential categorical variables, and normalisation for unbounded variables.

Geographical Coordinate System Transformation

Both the flightplan and actual trajectory are provided in a spherical coordinate system, with latitude (ϕ) & longitude (λ) in degrees and height (h) in feet. Neural networks are very sensitive to different data formats, and many studies have shown that it is necessary to provide Cartesian (x, y, z) coordinates as input (e.g. Overkamp[43] & Tran et al. [57]). The chosen reference frame will be the ENU reference frame with respect to the local tangent on the sphere. Where North, East and Up are positive. To make this transformation, first the spherical coordinates must be expressed in the ECEF reference frame. This is done with Equation 6.3 and Equation 6.4. In this equation a is the equatorial earth radius and b is the polar earth radius. Then the tangent can be determined and the coordinates can be expressed in the local Cartesian reference frame. This is done with Equation 6.5. For this transformation a reference (X_r, Y_r, Z_r) location is required. Because the sequence generation task spans three hours of flight, the flat surface of the ENU reference frame will give errors if it is placed on a fixed reference location. It is therefor decided that the reference will be at the aircraft location for each sample and that the conversion is made with every data-point in the sequence. This is a similar strategy as applied by Tran et al.[57]. Alternatively, Amsterdam Airport can be selected as the reference location since almost all flights converge there. A schematic overview of the reference frames is shown in Figure 6.1[3].

$$\begin{aligned} X_c &= N(\phi) + hcos\phi cos\lambda \\ Y_c &= N(\phi) + hcos\phi sin\lambda \\ Z_c &= N(\phi) + hsin\phi \end{aligned} \tag{6.3}$$

$$N(\phi) = \frac{a^2}{\sqrt{a^2 cos^2\phi + b^2 sin^2\phi}} \tag{6.4}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -sin\lambda_r & cos\lambda_r & 0 \\ -sin\phi_r cos\lambda_r & -sin\phi_r sin\lambda_r & cos\phi_r \\ cos\phi_r cos\lambda_r & cos\phi_r sin\lambda_r & sin\phi_r \end{bmatrix} \begin{bmatrix} X_c - X_r \\ Y_c - Y_r \\ Z_c - Z_r \end{bmatrix} \tag{6.5}$$

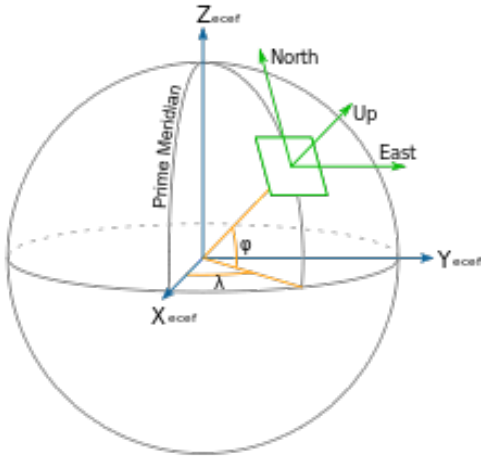


Figure 6.1: Different reference frames used for geographic positions: Yellow reference frame is the spherical coordinate system. Blue is the ECEF reference frame. Green is the Cartesian ENU reference frame.

[3] https://commons.wikimedia.org/wiki/File:ECEF_ENU_Longitude_Latitude_relationships.svg

Weather data matching

In order to include the weather data in the model, there are multiple methods which are to be tested for effectiveness. As discussed, the sequential nature of the input data may force an approach similar to Liu and Hansen[33] for the wind and temperature matching. Here, a matching algorithm is made that looks up the weather parameters in a grid up to 120km in front of the aircraft. Thereby effectively including a large region with every prediction. The downside of this may be a relatively large input which slows down training. Nevertheless, the results of this method do show accurate responses. Another possibility is to have a separate neural network that is not relying on sequential data to first process and encode the entire meteorological dataset. The encoded information can then be passed to the decoder together with the encoded B2B data input. This approach is taken by Zhang et al.[67], who apply a convolutional neural network encoder block to make a weather feature input. Both approaches will be explored during the modelling phase of the research.

Feature engineering

Part of the data that can be used as feature is not sequential but stays constant for the whole flight. For example the origin, aircraft type, operator, or callsign do not change. These values are usually text based features, which are not easily processed by a neural network. This can be solved by categorising variables if the entire set of values is known and bounded. One-hot-encoding can be applied, which changes text into numerical vectors. For example if three airlines are present in the batch of flights: KLM, Scandinavian & Lufthansa, one-hot-encoding will assign a vector of length three with a binary value for the actual category. KLM will be represented by the vector [1,0,0] and Scandinavian with [0,1,0]. The resulting variable is much easier to parse through a neural network and is called a dummy variable.

Data features that are not categorical cannot be encoded as dummy variables. However there may still be a correlation or dependency between the feature and the output. Variables that may become unbounded can have an undesirable impact on the weights in the neural network. For example delay is usually a low value, or only a couple of minutes. But if then a value of 8 hours is present in the dataset, this may return exploded results of the network. Normalisation techniques bound such values by analysing the entire dataset and scaling the values accordingly. Min-Max normalisation is most commonly used, which is shown in Equation 6.6. This can be applied to delay, taxi time, altitude, wind-speeds, latitude and longitude.

$$x' = \frac{(x - x_{min})}{(x_{max} - x_{min})} \tag{6.6}$$

In a sequence generation task, time is arguably the most important feature. Extra care must be taken in deciding how to parse timestamps and other time information. Not only is it important to concisely use the UTC time, it is also important that trajectories and flight-plans are properly comparable. In this research, the ground trajectory is placed out of scope. This is partly because the speeds and performance of the aircraft are considerably different from the airborne part. Also, the calculated take-off time is often significantly different from the actual take-off time. The goal is to build a TP that can model the deviations in the airborne trajectory and therefor the flight duration of the flightplan and the actual trajectory are probably more valuable. For this reason, the take-off time must be filtered from the trajectory and this marks the start of the sequence. Respectively so for the flight-plan. The timestamps will then be converted to duration timestamps. However time information may still convey a lot of information, as a midnight flight usually experiences less traffic and different ATC procedures then during the day. To include this information, take-off and FIR arrival time are still valuable parameters. These can be encoded cyclically to make sure that for the network, 23:59 is observably close to 00:01 the next day. This can be done with Equation 6.7, where H is the hour. Note that both a sin and cosine are required to avoid equivalency of mirrored times on the sinusoidal period.

$$\begin{aligned} H_{sin} &= sin(\frac{2\pi H}{max(H)}) \\ H_{cos} &= cos(\frac{2\pi H}{max(H)}) \end{aligned} \tag{6.7}$$

In conclusion, the data that will be used as features to the transformer neural network may need editing before successful implementation. It is important that these processes are reversible, as the output of the model will be in the same coordinate system and order of magnitude. Table 6.3 gives a general overview of the feature engineering applied before modelling start.

Table 6.3: Overview of all input features and the required feature engineering. Potentially other variables may be added during the modelling phase of the research.

Time Series Feature	Unit	Required feature transformation
FPL/ADS-B duration timestamp	seconds	-
FPL/ADS-B latitude	deg	Min-Max normalisation Coordinate system transformation
FPL/ADS-B longitude	deg	Min-Max normalisation Coordinate system transformation
FPL/ADS-B altitude	ft	Min-Max normalisation
V wind component	m/s	Min-Max normalisation
U wind component	m/s	Min-Max normalisation
Vertical air velocity	Pa/s	Min-Max normalisation
Air Temperature	K	Min-Max normalisation

Static Feature	Unit	Required feature transformation
Estimated take-off time	UTC	Cyclical Encoding
Day of week	-	One-Hot-Encoding/Cyclical Encoding
Month	-	One-Hot-Encoding/Cyclical Encoding
Pre-departure delay	seconds	Min-Max normalisation
Origin airport	-	One-Hot-Encoding
Aircraft Type	-	One-Hot-Encoding
Aircraft Operator	-	One-Hot-Encoding

6.4.3. Data analysis

With the available datasets, a preliminary estimation can be made on the current demand error with the methods based on planned trajectories. For this purpose a simple demand model is build with the actual FIR entry times obtained from the ADS-B trajectories, and the those from the filed flight plans. The arriving flights are aggregated in 20 minute interval bins, which is the standard bin size in current demand forecasts. Because almost all traffic entering the Amsterdam FIR is descending towards Schiphol airport, the entry flow is assumed to be the actual demand of the airspace. This differs from Upper Area Sectors, where occupancy duration can change significantly between flights. In lower airspace the occupancy is different mainly when holding is practised. Having said that, holding is usually a result of excessive demand and thereby a failed balance between demand and capacity. As a result, the assumption of entry flow as demand predictor can be applied to give an initial estimate. With this in mind, an example of a demand forecast for one of the days in the dataset is shown in Figure 6.2.

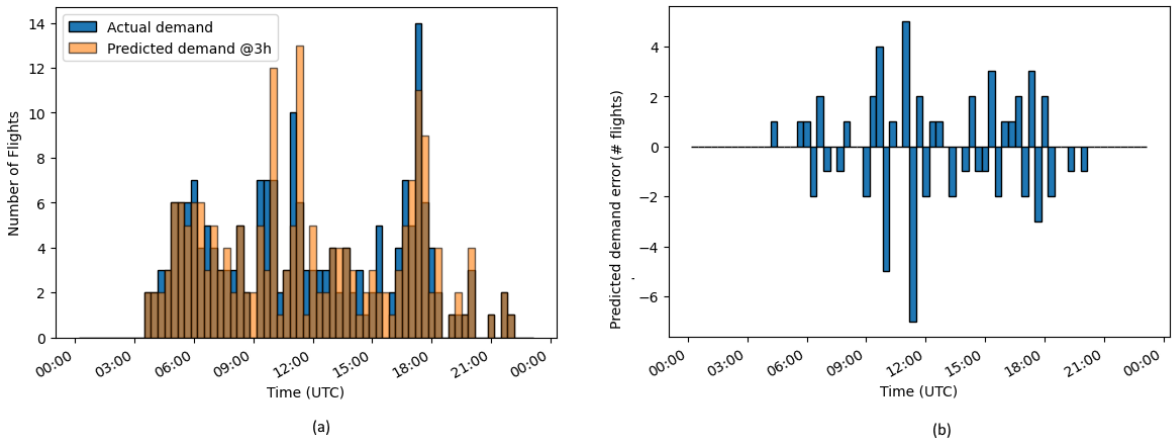


Figure 6.2: (a) Predicted and actual demand of the Amsterdam FIR at the 7th of may 2021. The predicted demand with a look-ahead time of 3 hours is shown. (b) Error between predicted and actual demand.

Looking at the demand errors, it becomes clear that part of the error is a result of flights arriving just in another bin then initially expected. This is clearly shown in Figure 6.2(b), where over- and under-predictions alternate. The highest errors are visible in the demand peaks, where the model predicts the traffic to arrive in a bunch at 10:00 and 11:00 UTC. This over prediction would lead to regulations and delays, which is undesirable. When trying to find a cause for such errors, it is expected that the actual take-off time is a significant element. Different studies have shown this is the case, amongst which the work of Könnemann[26], and research by LVNL[40]. Looking at the FIR arrival times for three randomly selected days in the dataset without high delay figures; a Root Mean Squared Error of 23 minutes is observed. With the error defined as the difference between the actual trajectory and flight plan arrival time. When the take-off time is known, a straightforward operation may be to add the ground delay to the arrival time of the flight plan. Doing this yields an RMSE of 17 minutes on the arrival times. The results for different look ahead times are shown in Figure 6.3. The blue line shows the regular root mean squared error of the arrival time, and the orange line shows those with the flight plans shifted with departure delay. Clearly, the error reduces the uncertainty for every look ahead time, which is in line with the analysis made by Könneman[26]. The measured standard deviation for the errors is shown with the vertical bars.

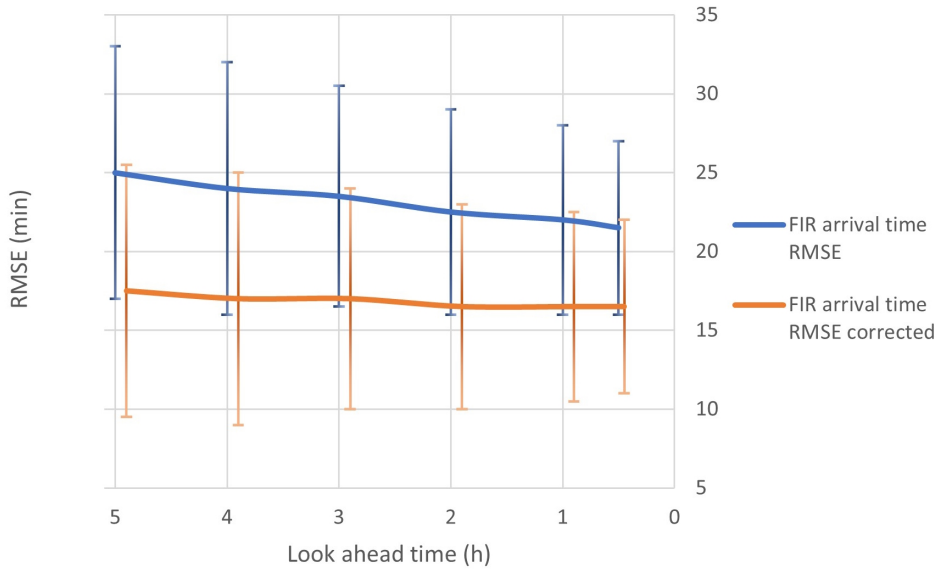


Figure 6.3: The error between actual and flight plan predicted FIR arrival time. The orange line shows the error of the predicted arrival time with the departure delay added. This is significantly lower, which proves that departure time uncertainty is an important source of error. The standard deviation is shown with the vertical bars.

The results mentioned above prove that demand prediction errors are partly due to uncertainty in take-off times. Nevertheless, Tielrooij et al.[55] have pointed out that delayed flights will usually fly faster to compensate for time lost. A simple delay time-shift of the trajectory is not an independent transformation. When evaluating the errors of days in the dataset that do have high departure delay, the correction produces errors that are statistically similar to the non-corrected predictions. In this research, the departure time uncertainty is placed out of scope. However, these results confirm that other errors, amongst which TP errors, remain a significant element of the total demand error. The remaining portion of arrival time error as presented in Figure 6.3 may be due to TP shortcomings.

Trajectory prediction error can be calculated by evaluating the altitude, along-track and cross track error. For the same datasets as the previously discussed demand errors, this is shown in Figure 6.4. The error is calculated for the estimated time overhead each waypoint and the actual position at that timestamp. Clearly the along track error becomes increasingly uncertain for longer look ahead times, which is to be expected. Cross track errors remain relatively constant because the direction of the flight does not change too much; the destination and therefor general track is constant. Similarly, the altitude error is also relatively constant, as the most efficient cruise altitudes and constrains do not change significantly.

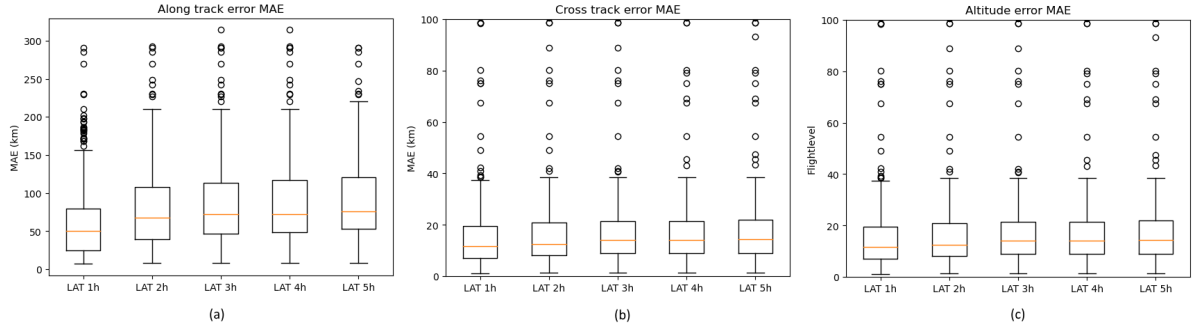


Figure 6.4: Error box plots for the (a) along track error, (b) cross track error, & (c) altitude error. The error is evaluated between the flight plan waypoints and the actual trajectory, based on the actual timestamp.

The error metrics in Figure 6.4 are evaluated based on the filed overhead times of the waypoint. This includes pre-departure delay, hence it does not reflect the true performance of the TP that calculates the flight profile from the flight plan. The true TP modelling capabilities can be evaluated by changing the time evaluation to a duration evaluation. When both the flight plan and actual trajectory start at t=0, the timestamps can be used to directly measure positional and altitude error. Doing this, a similar analysis can be made in Figure 6.5. The cross track and altitude error and variance do not change significantly. However, the along track error does show different behaviour. A significantly smaller error increase is found with look ahead times that are further out. The reason for this can be two folded. Either the flight plan based trajectories have a reasonably constant predictive performance, and/or the flight plans are not updated frequently. The latter is a relevant finding, as it was expected that NM would distribute updated information on flight position and aircraft intent in the tactical phase through the flight plan based trajectory. However in this specific dataset, this appears not to be the case. Arguably, this may make the use of B2B data non-unique compared to other flight plan datasets. However, if in the future the planned trajectory is updated, the TP improvements can be significant. Hence it is still a relevant data source. Moreover, the errors observed in Figure 6.5 are large enough to contribute significantly to demand errors. This shows that trajectory prediction indeed requires improvements.

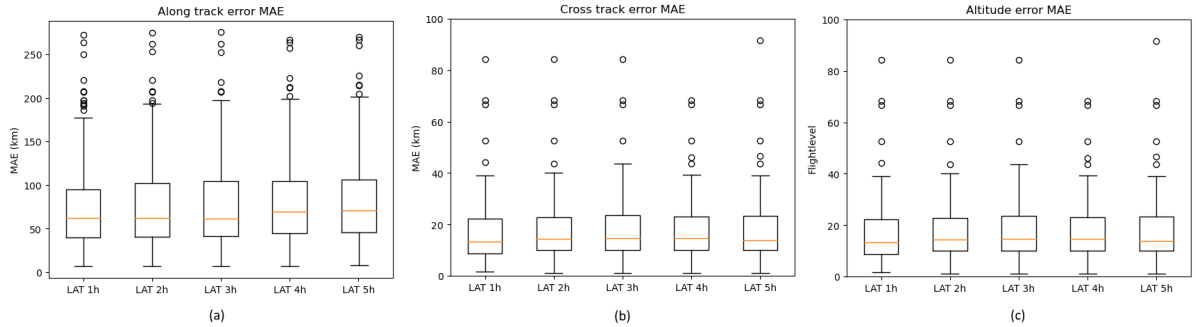


Figure 6.5: Error box plots for the (a) along track error, (b) cross track error, & (c) altitude error. The error is evaluated between the flight plan waypoints and the actual trajectory, based on the flight duration timestamp.

Examining a random flight in the dataset and visualising the trajectory may prove the error in more detail. Figure 6.6 shows an example of a flight from Malaga to Amsterdam on may 6th 2021. The red line is the filed flight plan at 3 hours before arrival in the FIR. The blue line is the actual flight trajectory obtained from ADS-B data. Two main differences are observed: In the lateral profile, the flightplan contains waypoints that are offset from the great circle (shortest track). This is primarily due to routing via airways which are not exactly aligned with the shortest route. Also, there may be military airspaces which are expected to be closed, so the flightplan must file around these sectors. The true trajectory shows that the flight did not entirely adhere to the flightplan. In central Spain the flight was allowed to fly direct towards Bordeaux. Also above France a couple of corners were cut. This may result in an early arrival.

In the vertical profile, the most obvious difference is the change of cruise altitude from flightlevel 400 to flightlevel 380. This may not have a direct influence on arrival time, but the cruise speed and resultant ground speed are affected by it. Planned cruise speed information is not retrievable from the current B2B messages dataset, so these performance differences cannot be evaluated. However, the final result of trajectory differences between planned and actual are highlighted by the error mark in Figure 6.6(b). The arrival time error due to TP error is found to be 4 minutes for this flight. That may seem like a relatively small error on the scale of 3 hours look ahead. However, for Schiphol the traffic demand in the peak hours can be on the edge of the available capacity. A 4 minute early arrival, or a delay of a couple of minutes may just be enough to issue a regulation. Especially when this error is present in a large number of flights. Referring back to Figure 6.3, the total error when using only flight plan based trajectory predictions is still relatively high. A gain of a couple of minutes on the error can mean a lot to the predictability of demand.

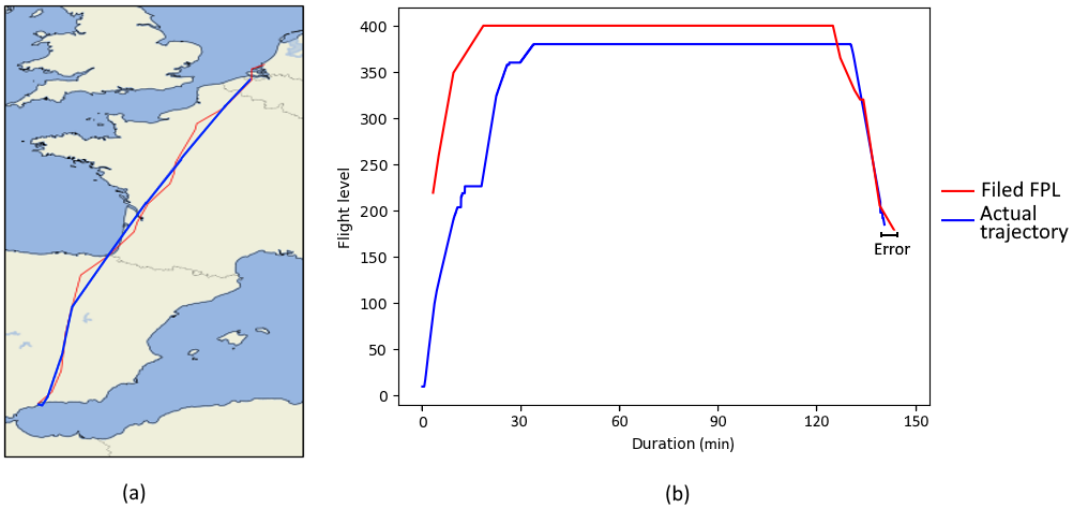


Figure 6.6: Visualisation of the actual 4D trajectory (blue) and the filed flight plan (red) of flight on may 6th 2021. (a) Lateral flight profile, (b) Vertical flight profile.

Another interesting disadvantage of relying on the flight plan is that entry locations of the flights are fixed on airways or arrival routes. In reality, deviations, runway changes, and vectoring can change the entry points significantly. The arriving flights of a single day in the dataset are shown in Figure 6.7(a). In Figure 6.7(b) the respective flightplan entries are shown. This figure clearly shows the degree to which the actual flights differ from filed flight plans.

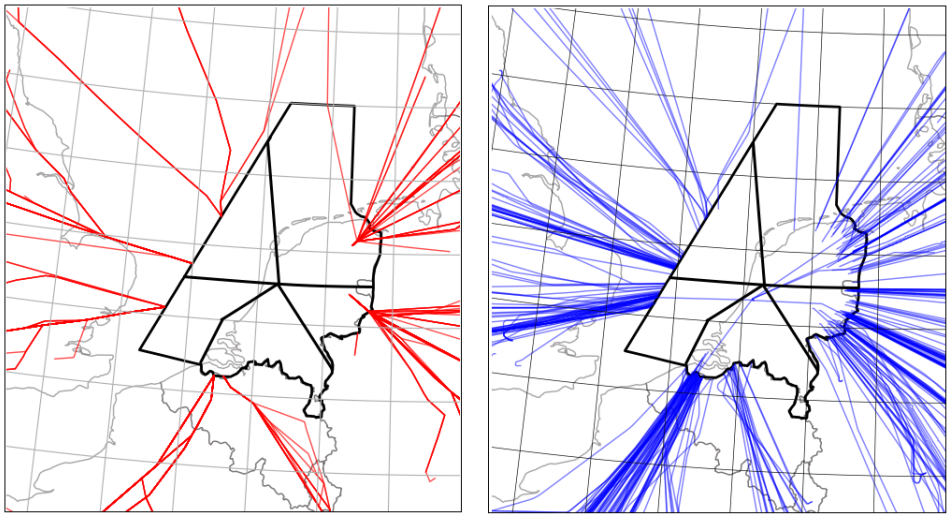


Figure 6.7: (a) Filed flight plans up to FIR entry on may 15th 2021, with a look ahead time of 3 hours. (b) Actual flights up to FIR entry on may 15th 2021

Flights that have a different entry point but still have an accurate arrival time may not be a big problem, given that the FIR is a single sector. In reality, the FIR is subdivided into different blocks of airspace. For this research, the sectors that are of importance are the area control centre sectors. The upper part (FL195-FL245) of the Dutch airspace is divided into five different sectors that are controlled by LVNL. The sectors are originating from the SPY VOR beacon and denoted by the black areas in Figure 6.7. If flights have a different entry point then planned, this may influence the demand forecast if the change results in a different entry sector. Furthermore, a different entry point or altitude will also influence the sector occupancy time. Both these effects can be improved with a data driven trajectory predictor such as the transformer neural network.

7

Model Development & Analysis

7.1. Development of predictive models

The next phase of the research is to start building predictive models. The chosen methodology is to first build a trajectory predictor based on the transformer neural network architecture. This model then provides the entry point and entry time of flights to the subsequent demand predictor model. A simple iteration of a demand predictor was already build to analyse demand forecasting performance with the methodology applied in current operations. The baseline results have been discussed in subsection 6.4.3. Both predictive models must be developed towards a comparative experiment, which is explained in the following sections.

7.1.1. Trajectory predictor development

As discussed previously, the demand forecast error observed in operations currently is often too uncertain due to trajectory predictor error. Although efforts have been made to bypass a TP entirely, novel machine learning models may be able to deal with some of the limitations in conventional trajectory prediction. Most importantly the ability to capture alternative routes as shown by Liu et al.[33] amongst others is promising. This research proposes a deep generative trajectory prediction approach to improve the demand forecasting. The selected methodology will be the transformer neural network that was introduced by Vaswani et al.[58].

Building a machine learning model is an iterative process, due to the large amount of parameters that can be specified. The number of layers, layer size, number of attention heads, learning rate, cost function and optimiser are some of the many variables. The performance of the model is highly dependent on these parameters and the training that is performed. A clear strategy to obtain an optimised model does not exist. Nevertheless a development roadmap is set up to provide a step-wise modelling approach. First of all, a single cluster of flights, or origin region is selected that has enough flights to begin modelling. This could for example be flights from Spain to Amsterdam. In this way the solution space is kept within bounds. The model building begins with a very simple model, both in terms of input features and in layer complexity. First, the input features will only be the 4D sequences of the flight plan and actual trajectory. The initial model will only contain a few layers with a default number of attention heads (8). The model can then step-wise be expanded with another layer or another feature from the available dataset. Other parameters will be specified on an iterative basis. Furthermore, overfitting measures such as batch training and dropout are applied in this step as well.

Once the first simple model iteration has reached a satisfactory level of detail, the model can be extended to include different groups of flights. Initially the results for a diverse set of flights will be tested to see if the transformer neural network is capable of capturing the different characteristics. If this is not the case, clustering may be required to group flights. DBSCAN is a proven clustering technique and will therefor be applied if necessary. Consecutively, different models will be trained for each cluster.

After the extension of the model to different city pairs, post processing methods will be implemented. Different studies have shown that it is not a straightforward operation to extract smooth trajectories from a neural network. The output is a set of Gaussian mixtures, where the attribute with the highest

probability is usually taken as the prediction. If the variance is too high, this may return zig-zag output trajectories. Kalman filtering, log likelihood calculation or other smoothing algorithms may be required to infer a smooth output trajectory. When the demand predictor is build on probabilistic basis, the Gaussian mixtures output may be valuable. In this case the smoothing may not be required, but this is yet to be investigated.

Depending on the time and success of the conceived model, alternatives to a transformer with encoder-decoder structure may be worthwhile exploring. If sufficient time remains, the R-Transformer network developed by Wang et al.[60], will be on the shortlist to develop. This model combines a local recurrent neural network with multi-headed attention layers, which may be able to better capture local patterns within the feature sequences. If this model will be developed, a similar development strategy as mentioned above will be applied. Finally, it must be noted that the machine learning models will be trained on a dataset with a sufficient amount of flights, which are representative for normal operations in the Dutch FIR. The size of the dataset will be determined based on the first modelling efforts. This also depends on how much of the dataset will be used for training, testing and validation.

7.1.2. Demand predictor development

The demand predictor module of the research is not the most complex part, but nonetheless important. For initial data analysis, a simple model was already developed that predicts demand by calculating the arrival time in the FIR from the flight plan. This information is aggregated in bins of 20 minutes, which is effectively the sector entry flow. Because the demand in this specific situation is directly dependent on the arriving traffic flow, the assumption holds that sector demand may be modelled with arrival times. In the further stages of the research this demand predictor logic is to be extended to calculate actual demand. Furthermore, if sufficient time remains, a different demand forecasting model may be developed to compare the performance of the TP based model with an alternative demand model.

To further develop the demand forecasting model, there may be different paths that can be chosen. Before selecting an approach, it is important to keep in mind the problem statement. Namely to improve the demand forecast with machine learning based trajectory prediction. Given that this method is already applied to the flight segment before sector entry, no real necessity remains to extend this into the demand forecasting. Henceforth, the demand forecasting module should be consistent for each method such that the results are comparable. In order to evaluate demand instead of sector entry flow, one possibility is to calculate sector exit times with statistical records on the basis of e.g. aircraft type, runway usage, aircraft operator amongst others. This would yield equivalent results for every TP driven method. Another possibility that will be explored is to obtain the sector exit time by extending the TP calculation through the sector. However, due to the large amount of vectoring applied by air traffic control, the results are expected to be of lower quality.

Once the demand prediction module is completed and the planning allows it, another methodology may be explored to compare results in the experiment. Sector flow models such as the one developed by Menon et al.[37] are likely too extensive for the scope of this research. A probabilistic model on the other hand may be feasible to implement. Such a model can be applied relatively straightforward if the Gaussian mixtures of the trajectory predictor are available. A probabilistic model then aggregates the probabilities of aircraft demand based on the uncertainty of arrival time in the TP. This is less straightforward for the conventional flightplan based methodology, but Gilbo et al.[21] have shown that this can be bypassed by modelling the arrival times of aircraft as a Gaussian distribution. Last but not least, the current demand predictor in the LVNL decision support tool also has a corrective model based on a random forest regressor. The results of this model may be compared with the demand forecast of this research.

7.2. Experiment set-up

Once the trajectory predictor and demand predictor have been developed to a satisfactory level, an experiment can be designed to test the model and answer the research questions. As mentioned, the baseline model will be an auto-encoder transformer neural network that has a flightplan sequence as input, and generates a 4D trajectory up to the Dutch FIR. The trajectory is then used as input to the demand predictor. The performance of the model is subsequently tested and compared to the actual demand from the recorded ADS-B trajectories. Nevertheless, different models are developed, which

include different features or have different architectures. The experiment will therefor be tasked with comparing the different results and finding the optimal combination of features, model complexity, and architecture. Currently the experiments listed in Table 7.1 are foreseen.

Table 7.1: Planned experiments to evaluate the performance of the different model varieties and the effect of different input features. Baseline experiment 0 was already developed in chapter 6.

Experiment	Model	Features
0 (baseline)	TP: Flight plan Demand: Basic TP based model	Flight plan
1	TP: Simple Transformer Network Demand: Basic TP based model	Filed flight plan, ADS-B trajectories
2	TP: Extensive Transformer Network Demand: Basic TP based model	Filed flight plan, additional flight info & ADS-B trajectories
3	TP: Extensive Transformer Network Demand: Basic TP based model	Filed flight plan, additional flight info, Weather & ADS-B trajectories,
4	TP: Best model from exp. 1- 3 Demand: Probabilistic TP based model	Best performing feature set from experiment 1- 3
5	TP: N.A. Demand: Simple machine learning aggregator (e.g. Random Forest)	Planned ETA at FIR boundary

As mentioned before, building a machine learning model is an iterative process, and along the way new insights may be found. Potentially this can lead to a different schedule of experiments or a slight change of model and feature selection. However, the plan is to stick to the philosophy of the listed experiments in Table 7.1. In the following phase of the research, the models will be build and the experiments can be worked out in more detail. Exact feature specification and model architecture are to be determined, after which the actual experiments can be performed.

7.3. Results & Analysis

The objective of this research is to improve air traffic sector demand forecasting in the tactical domain, by exploring machine learning based trajectory prediction. The experiments have been designed in such a way that it can answer the research question and hopefully meet the objective. In this literature study the first sub questions of the research question have already been answered. For example the transformer neural network is derived from literature as the most promising and suitable machine learning model for the TP task (subquestion 6). Also, in chapter 6, the most relevant features and parameters have been identified and processed for the demand and trajectory prediction models (subquestion 1&4). What remains to be answered is the performance and applicability of the different TP models on demand forecasting for air traffic sectors in the Dutch FIR. The performance will be evaluated on the demand predicting capabilities. Both the prediction accuracy and uncertainty will be evaluated, based on a large enough validation dataset. This dataset will never be used for training or testing the model, hence is completely independent. The accuracy can be calculated by error metrics such as the Root Mean Squared Error, Mean Absolute Error or R^2 value. The uncertainty of the predictions will be measured by calculating the variance and standard deviation of the distribution. The standard deviation is used to compute a 95%-confidence interval. The higher the spread of the error distribution, the more uncertain the predictions and the wider the confidence interval. This way, the performance of the different experiments can be evaluated and compared to the baseline.

Although demand forecasting is the primary goal of the research, this research is a result of the general movement within Air Traffic Management to move towards a trajectory based operation. The predicted trajectories are equally important to the academic field. For this reason the predictive accuracy and uncertainty of the TP models will be extensively reviewed as well. The along track, cross track, horizontal and vertical error will be evaluated with similar metrics as the demand accuracy and uncertainty. If during the following phases of the research different results are found to be of value, these will be disclosed as well.

8

Research Planning

This research project is structured into four distinct phases. An overview of the phases and the more detailed work out of this is shown in Figure 8.1. The main phases are:

1. **Literature study:** The first stage of the research is to identify the problem, formulate a research objective, and describe a research question. This is done through studying and reporting on related academic work and operational documents. This phase is completed and documented in this report.
2. **Data collection and preparation:** During this stage of the research, all required data is gathered and analysed to support the research question and literature findings. Also during this phase, a strategy for model development is worked out. This phase is completed and documented in this report.
3. **Model development:** The following phase includes all the work that is to be done in order to run the experiments that are required to answer the research question. The model is developed based on the strategy worked out in phase 2.
4. **Results and analysis:** The final step in the research is to perform the experiments and to analyse the results. The different models & experiments are compared, and conclusions can be drawn.

The first two phases of the research project have been finished, with the literature study and data preparation completed. The next phases are to develop the different models and set up the experiments. Once the experiments are performed, the results can be analysed. The dataset derived from the initial data processing will be used to build the different models and experiments. If during the following phase a more suitable dataset will be made available, then this will be implemented instead. A detailed planning is presented in Figure 8.1, which serves as a guideline for the remainder of the research project.

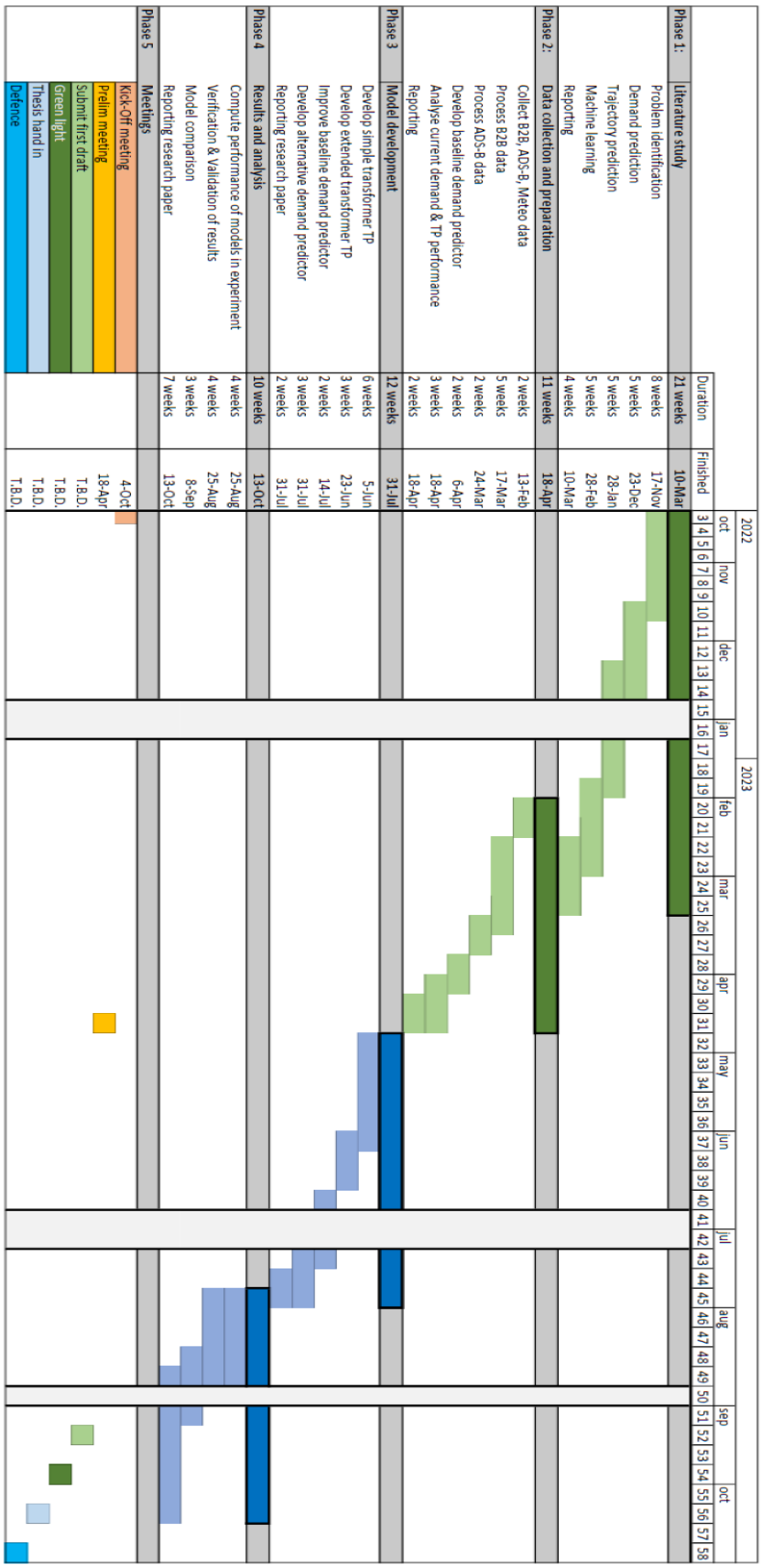


Figure 8.1: Gantt chart with the current planning of the research activities. Green activities have been completed and blue activities are scheduled. The vertical white bars mark the holidays during which no work is scheduled.

9

Conclusion

In current Air Traffic Management operations, many processes contribute to a safe and efficient execution of flights. Demand and capacity balancing is an important element to ensure that controlled airspaces are not overloaded with traffic. LVNL, the Dutch ANSP, introduced a decision support tool that makes a forecast of the expected traffic load in the coming hours. Simultaneously, this tool provides options to manage excessive traffic flows to the air traffic controller supervisors. Although this tooling gives a reasonably accurate forecast, significant errors may still be observed that cause unnecessary or ineffective interactions. For example, imposing delays on aircraft that are yet to depart. Delays are undesirable, especially if the delay was not required in hindsight. The demand prediction system currently in place is based on flight information received from the Eurocontrol Network Manager, which does not always yield the best results. Better predictions of demand can increase the effectiveness of the demand & capacity balancing process. Therefor this research objective is to improve air traffic sector demand forecasting in the tactical domain, by exploring machine learning based trajectory prediction. The research question that is asked to reach the objective is the following: To what extent can machine learning be applied to long term trajectory prediction, and how can its output contribute to demand forecasting of a particular air traffic sector?

First, to answer the research question, demand forecasting was investigated in chapter 3. Nowadays, operational systems rely on flight plans or trajectories that originate from a central provider, such as the Eurocontrol ETFMS or the FAA TFMS system. Although flight plans give a basic indication of the flight execution, it still contains too much uncertainty for reliable demand predictions. The departure time of flights is a big source of uncertainty, and actual flights deviate from the plan very often. Different improvements have been proposed, amongst which a probabilistic demand predictor, or Hidden Markov Model, to optimise trajectories for demand. Trajectories however are not the only way to forecast demand. Aggregate demand prediction does not specify individual flight trajectories, but rather models a whole network of sectors that interact via flows. This is called a sector flow model, and only general flight information is required. Various research efforts have proposed these models and proven them to be more effective at long term demand predictions than flight plan based models. Nonetheless, this research will focus on demand prediction through trajectories. This method is selected primarily because of the general ATM development direction towards a Trajectory Based Operation.

Moreover, based on the chosen direction of a trajectory based approach, chapter 4 explores the trajectory prediction methodology. Two different classes have been identified: Model based or data driven trajectory prediction. Conventionally, trajectory prediction is based on an aircraft performance model, which contains performance parameters such as cruise speed, climb rates or fuel consumption for different aircraft in a variety of flight conditions. The trajectory is derived from extrapolating an initial state & intent, via a behaviour model and a mathematical model. This method is used in most of the current demand forecasting applications. The second class of models is data driven trajectory prediction. This type of model relies on big datasets containing trajectories, and tries to apply a mathematical model to describe samples in the dataset. Most of these models are machine learning algorithms that generate or extrapolate a trajectory based on previous aircraft states. Hidden Markov Models or varieties of recurrent neural networks have been widely adopted in academic research. Drawing comparisons

between the two methodologies, there is no clear winner. This is primarily due to the large variety of application. When choosing a method, the application and objectives are very important. This research focuses on the long term generative trajectory prediction. On long term predictions, model based TP is limited by a lack of intent information. Data driven approaches may bypass this by statistical pattern recognition. Therefor, a data-driven approach is selected.

Third, in chapter 5, machine learning models that are relevant to demand forecasting or trajectory prediction are discussed. Looking at the latest academic research, most of the developments focus around neural networks. For demand forecasting, the convolutional neural network and graph neural network were discussed, where especially the latter has shown significant improvements. In trajectory prediction, clustering and recurrent neural networks such as the LSTM network are amongst the latest developments. Clustering can either be applied to predict a global trajectory, or to group flights for more effective modelling with a subsequent TP. The LSTM neural network was used by different researchers to generate trajectories pre-departure. These results show the feasibility and potential gains of such models within the three hour look ahead time that is of interest to this research. Nonetheless, there are different models that may be promising but are yet to be applied on the trajectory prediction domain. Especially the transformer neural network is found to be a promising methodology. In different sequence-to-sequence modelling tasks, such as language processing, this method has proven to converge much faster with equal or better results. For this reason, the transformer neural network will be developed to generate trajectories in support of demand forecasting.

Moving on,chapter 6 explains the available data and required pre-processing in detail. This research is a collaboration between the TU Delft and KDC mainport Schiphol. For this reason, the area of interest is the Netherlands airspace. The available data consists of flight information messages from the Eurocontrol B2B services to LVNL, and actual ADS-B trajectories from OpenSky. The B2B dataset contains flightplan and general flight information. The ADS-B data contains the actual 4D flight trajectories of flights that arrive to the Netherlands airspace. Additionally, weather data from the ECMWF ERA5 model may be used in further model development. The dataset used for analysis is from may 2021. A simple demand predictor model was build, which was used to quantify current prediction errors. Also the errors between actual and flight plan based trajectories were evaluated. These insights serve as a baseline for further research. Lastly, required data pre-processing for the transformer neural network was discussed.

Finally, chapter 7 discusses the proposed model development. The development strategy is largely dependent on available time and resources, but primarily focuses on first building a simple transformer neural network for trajectory prediction. This will then be extended with additional data features and model complexity. The demand predictor will be extended from entry flow to complete demand prediction. When time allows, a simple machine learning aggregate model may be build to compare the performance of different methods. The goal of the following phase is to conduct and analyse the five experiments that have been designed. To conclude, chapter 8 gives a planning and overview of the remaining two phases of the research.

References

[1] R. Alligier, D. Gianazza, and N. Durand. “Learning the aircraft mass and thrust to improve the ground-based trajectory prediction of climbing flights”. In: *Transportation Research Part C: Emerging Technologies* 36 (2013), pp. 45–60. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2013.08.006. URL: https://www.sciencedirect.com/science/article/pii/S0968090X13001708.

[2] J.D. Anderson. *Introduction to Flight*. McGraw-Hill Higher Education. McGraw-Hill Higher Education, 2016, p. 119. ISBN: 9789814636186.

[3] *ATFCM users manual*. 16th ed. EUROCONTROL, Brussels, Belgium. 2012.

[4] Samet Ayhan and Hanan Samet. “Aircraft Trajectory Prediction Made Easy with Predictive Analytics”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 21–30. ISBN: 9781450342322. DOI: 10.1145/2939672.2939694. URL: https://doi.org/10.1145/2939672.2939694.

[5] Luis Basora, Jérôme Morio, and Corentin Mailhot. “A Trajectory Clustering Framework to Analyse Air Traffic Flows”. In: *SID 2017, 7th SESAR Innovation Days*. Belgrade, Serbia, Nov. 2017. URL: https://hal-enac.archives-ouvertes.fr/hal-01655747.

[6] Alessandro Bombelli, Adrià Segarra Torné, Eric Trumbauer, and Kenneth D Mease. “Improved clustering for route-based eulerian air traffic modeling”. In: *Journal of Guidance, Control, and Dynamics* 42.5 (2019), pp. 1064–1077.

[7] Dan Chen, Minghua Hu, Yuanyuan Ma, and Jianan Yin. “A network-based dynamic air traffic flow model for short-term en route traffic prediction”. In: *Journal of Advanced Transportation* 50.8 (2016), pp. 2174–2192. DOI: https://doi.org/10.1002/atr.1453. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/atr.1453. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/atr.1453.

[8] Ramon Dalmau Codina, Marc Melgosa Farrés, Santi Vilardaga García-Cascón, and Xavier Prats Menéndez. “A Fast and Flexible Aircraft Trajectory Predictor and Optimiser for ATM Research Applications”. In: 2018.

[9] Rod Cole, Steve Green, Matthew Jardin, Barry Schwartz, and Stanley Benjamin. “Wind Prediction Accuracy for Air Traffic Management Decision Support Tools”. In: (Feb. 2000).

[10] Council of European Union. *Council regulation (EU) no 1207/2011*. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:02011R1207-20200520. 2020.

[11] Council of European Union. *Report from the commission to the European Parliament and the council on the implementation and progress of the Single European Sky during the 2012-2014 period (COM(2015) 663 final)*. https://transport.ec.europa.eu/system/files/2016-09/com%25282015%2529663.pdf. 2015.

[12] Casper Dek. “Predicting 4D trajectories of aircraft using neural networks and gradient boosting machines: A data-driven aircraft trajectory prediction study”. In: *TU Delft Repository* (June 2020). URL: http://resolver.tudelft.nl/uuid:0b5b941e-6e25-46fa-bfb8-c2f5eaeddd23.

[13] Daniel Delahaye, Adrían García, Julien Lavandier, Supatcha Chaimatanan, and Manuel Soler. “Air Traffic Complexity Map Based on Linear Dynamical Systems”. In: *Aerospace* 9.5 (2022). ISSN: 2226-4310. DOI: 10.3390/aerospace9050230. URL: https://www.mdpi.com/2226-4310/9/5/230.

[14] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD’96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[15] G. Calis F. Dijkstra. *System Subsystem Specification (SSS) Decision Support Tool*. 0.9.10. Luchtverkeersleiding Nederland. 2020.

[16] Yann Le Fablec and Jean-Marc Alliot. “Using Neural Networks to Predict Aircraft Trajectories”. In: *International Conference on Artificial Intelligence*. 1999.

[17] Esther Calvo Fernández, José Manuel Cordero, George A. Vouros, Nikos Pelekis, Theocharis Kravaris, Harris V. Georgiou, Georg Fuchs, Enrique Casado, Pablo Costas, and Samet Ayhan. “DART : A Machine-Learning Approach to Trajectory Prediction and Demand-Capacity Balancing”. In: 2017.

[18] Eduardo Gallo, Javier Lopez-Leones, Miguel A. Vilaplana, F.A. Navarro, and Angela Nuić. “Trajectory computation Infrastructure based on BADA Aircraft Performance Model”. In: *2007 IEEE/AIAA 26th Digital Avionics Systems Conference* (2007), pp. 1.C.4-1-1.C.4–13.

[19] Maxime Gariel, Ashok N. Srivastava, and Eric Feron. “Trajectory Clustering and an Application to Airspace Monitoring”. In: *IEEE Transactions on Intelligent Transportation Systems* 12.4 (2011), pp. 1511–1524. DOI: 10.1109/TITS.2011.2160628.

[20] Mohammad Ghasemi Hamed, David Gianazza, Mathieu Serrurier, and Nicolas Durand. “Statistical prediction of aircraft trajectory: regression methods vs point-mass model”. In: June 2013.

[21] Eugene Gilbo and Scott Smith. “A New Model to Improve Aggregate Air Traffic Demand Predictions”. In: *AIAA Guidance, Navigation and Control Conference and Exhibit*. DOI: 10.2514/6.2007-6450. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2007-6450. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2007-6450.

[22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[23] Rik Graas. “Uncertainty Modelling in Aircraft Trajectory Predictions”. In: *TU Delft Repository* (July 2021). URL: http://resolver.tudelft.nl/uuid:45b2be80-c8df-4842-b1c7-02d1665fb64c.

[24] Wang Hongyong, Ziqi Song, and Ruiying Wen. “Modeling Air Traffic Situation Complexity with a Dynamic Weighted Network Approach”. In: *Journal of Advanced Transportation* 2018 (Jan. 2018), pp. 1–15. DOI: 10.1155/2018/5254289.

[25] ICAO. “Doc 4444–Procedures for Air Navigation Services: Air Traffic Management”. In: *International Civil Aviation Organization Montréal* (2016).

[26] Erik Könnemann. “Performance Impact of Improved Departure Time Prediction Relative to Sector Demand & Arrival Time Predictability”. In: *TU Delft Repository* (Dec. 2014). URL: https://repository.tudelft.nl/islandora/object/uuid:0db87565-1d65-4dfd-8a15-ac20482372f0/datastream/OBJ/download.

[27] Michael Konyak, Scott Doucett, Robab Safa-Bakhsh, Eduardo Gallo, and Paul Parks. “Improving Ground-Based Trajectory Prediction through Communication of Aircraft Intent”. In: *AIAA Guidance, Navigation, and Control Conference*. DOI: 10.2514/6.2009-6080. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2009-6080. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2009-6080.

[28] Wu Kun and Pan Wei. “A 4-D trajectory prediction model based on radar data”. In: *2008 27th Chinese Control Conference*. 2008, pp. 591–594. DOI: 10.1109/CHICC.2008.4605732.

[29] James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontañón. “FNet: Mixing Tokens with Fourier Transforms”. In: *CoRR* abs/2105.03824 (2021). arXiv: 2105.03824. URL: https://arxiv.org/abs/2105.03824.

[30] Arjen de Leege, Marinus M. Van Paassen, and Max Mulder. “A Machine Learning Approach to Trajectory Prediction”. In: AIAA Guidance, Navigation, and Control (GNC) Conference, Aug. 2013. ISBN: 978-1-62410-224-0. DOI: 10.2514/6.2013-4782.

[31] Michael James Lighthill and Gerald Beresford Whitham. “On kinematic waves II. A theory of traffic flow on long crowded roads”. In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 229 (1955), pp. 317–345.

[32] Yi Lin, Jian-wei Zhang, and Hong Liu. “Deep learning based short-term air traffic flow prediction considering temporal–spatial correlation”. In: *Aerospace Science and Technology* 93 (2019), p. 105113. ISSN: 1270-9638. DOI: https://doi.org/10.1016/j.ast.2019.04.021. URL: https://www.sciencedirect.com/science/article/pii/S1270963818315864.

[33] Yulin Liu and Mark Hansen. “Predicting Aircraft Trajectories: A Deep Generative Convolutional Recurrent Neural Networks Approach”. In: (2018). DOI: 10.48550/ARXIV.1812.11670. URL: https://arxiv.org/abs/1812.11670.

[34] Chunyao Ma, Sameer Alam, Qing Cai, and Daniel Delahaye. “Sector Entry Flow Prediction Based on Graph Convolutional Networks”. In: *International Conference on Research in Air Transportation*. 2022.

[35] Chunyao Ma, Sameer Alam, Qing Cai, and Daniel Delahaye. *Sector entry flow prediction based on graph convolutional networks*. National Research Foundation (NRF), 2022. URL: https://www.icrat.org/.

[36] Rodrigo Marcos, Oliva G Cantú Ros, and Ricardo Herranz. “Combining Visual Analytics and Machine Learning for Route Choice Prediction”. In: *Seventh SESAR Innovation Days, November 2017* (2017).

[37] Padmanabhan K. Menon, Gregory D. Sweriduk, and Karl Bilimoria. “New Approach for Modeling, Analysis, and Control of Air Traffic Flow”. In: *Journal of Guidance Control and Dynamics* 27 (2002), pp. 737–744.

[38] Larry Meyn. “Probabilistic methods for air traffic demand forecasting”. In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. 2002, p. 4766.

[39] Stephane Mondoloni, Sip Swierstra, and Mike Paglione. “Assessing Trajectory Prediction Performance – Metrics Definition”. In: Jan. 2005, pp. 3.C.1–31. ISBN: 0-7803-9307-4. DOI: 10.1109/DASC.2005.1563347.

[40] Luchtverkeersleiding Nederland. *Software Design Description (SDD), Decision Support Tool - Load Predictor*. 2.0. 2020.

[41] Xavier Olive. “traffic, a toolbox for processing and analysing air traffic data”. In: *Journal of Open Source Software* 4 (2019), p. 1518. ISSN: 2475-9066. DOI: 10.21105/joss.01518.

[42] Xavier Olive and Luis Basora. “A Python Toolbox for Processing Air Traffic Data: A Use Case with Trajectory Clustering”. In: Nov. 2019. DOI: 10.29007/sf1f.

[43] Jean-Luc Overkamp. “Long Short-Term Memory Network Based Trajectory Prediction Incorporating Air Traffic Dynamics”. In: *TU Delft Repository* (Sept. 2021). URL: http://resolver.tudelft.nl/uuid:15358485-1caf-48df-84c8-9f4c389a27fa.

[44] Yongzhen Arthur Pan, Mario A. Nascimento, and Joerg Sander. “Online Stochastic Prediction of Mid-Flight Aircraft Trajectories”. In: *Proceedings of the 12th ACM SIGSPATIAL International Workshop on Computational Transportation Science*. IWCTS’19. Chicago, IL, USA: Association for Computing Machinery, 2019. ISBN: 9781450369671. DOI: 10.1145/3357000.3366144. URL: https://doi.org/10.1145/3357000.3366144.

[45] Damir Poles, Angela Nuic, and Vincent Mouillet. “Advanced aircraft performance modeling for ATM: Analysis of BADA model capabilities”. In: *29th Digital Avionics Systems Conference*. 2010, pp. 1.D.1-1-1.D.1–14. DOI: 10.1109/DASC.2010.5655518.

[46] Mathijs Post. “Assessment of Inbound Air Traffic Flow Management Delay and Total Arrival Delay for Amsterdam Airport Schiphol”. In: *TU Delft Repository* (June 2021). URL: http://resolver.tudelft.nl/uuid:e28ce39a-8ddc-41c1-a752-d5a92b7f2efa.

[47] Bart Rozendaal. *A neural network approach in optimising airport strategy with trajectory prediction.* Tech. rep. Knowledge and Development Center Mainport Schiphol, 2022, pp. 1–36.

[48] Benjamin Sanchez-Lengeling, Emily Reif, Adam Pearce, and Alexander B. Wiltschko. “A Gentle Introduction to Graph Neural Networks”. In: *Distill* (2021). https://distill.pub/2021/gnn-intro. DOI: 10.23915/distill.00033.

[49] Banavar Sridhar, Gano Chatterji, Kapil Sheth, and Tarun Soni. “An Aggregate Flow Model for Air Traffic Management”. In: *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM* 29 (July 2006), pp. 992–997. DOI: 10.2514/1.10989.

[50] Banavar Sridhar, Neil Chen, and Hok Ng. “An Aggregate Sector Flow Model for Air Traffic Demand Forecasting”. In: *9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO)* (Sept. 2009). DOI: 10.2514/6.2009-7129.

[51] Junzi Sun. “Open Aircraft Performance Modeling: Based on an Analysis of Aircraft Surveillance Data”. English. PhD thesis. Delft University of Technology, 2019. ISBN: 978-94-6384-030-9. DOI: 10.4233/uuid:af94d535-1853-4a6c-8b3f-77c98a52346a.

[52] Junzi Sun. *The 1090 Megahertz Riddle: A Guide to Decoding Mode S and ADS-B Signals*. 2nd ed. TU Delft OPEN Publishing, 2021. ISBN: 978-94-6366-402-8. DOI: 10.34641/mg.11.

[53] Junzi Sun, Tristan Dijkstra, Constantinos Aristodemou, Vlad Buzetelu, Theo Falat, Tim Hogenelst, Niels Prins, and Benjamin Slijper. “Designing Recurrent and Graph Neural Networks to Predict Airport and Air Traffic Network Delays”. In: *10th International Conference for Research in Air Transportation*. FAA & Eurocontrol. 2022.

[54] M. Tielrooij. “Arrival Management Support in the Presence of Prediction Uncertainty”. English. PhD thesis. Delft University of Technology, 2022. ISBN: 978-94-6421-878-7. DOI: 10.4233/uuid:a403112b-48a2-40d7-9db3-a5b754f31eee.

[55] Maarten Tielrooij, Robert Kok, Theo de Jong, Ferdinand Dijkstra, Tim Dufourmont, Esther Lap, Ander Okina, and Reinier Vos. Tech. rep. Knowledge & Development Centre Mainport Schiphol, Feb. 2022. URL: https://kdc-mainport.nl/wp-content/uploads/2022/03/KDC-Transition-to-Trajectory-Based-Operations-Report-v1.0final.pdf.

[56] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. *MLP-Mixer: An all-MLP Architecture for Vision*. 2021. DOI: 10.48550/ARXIV.2105.01601. URL: https://arxiv.org/abs/2105.01601.

[57] Phu N. Tran, Hoang Q. V. Nguyen, Duc-Thinh Pham, and Sameer Alam. “Aircraft Trajectory Prediction With Enriched Intent Using Encoder-Decoder Architecture”. In: *IEEE Access* 10 (2022), pp. 17881–17896. DOI: 10.1109/ACCESS.2022.3149231.

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[59] Zhengyi Wang, Man Liang, and Daniel Delahaye. “A hybrid machine learning model for short-term estimated time of arrival prediction in terminal manoeuvring area”. In: *Transportation Research Part C: Emerging Technologies* 95 (2018), pp. 280–294. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2018.07.019. URL: https://www.sciencedirect.com/science/article/pii/S0968090X18310167.

[60] Zhiwei Wang, Yao Ma, Zitao Liu, and Jiliang Tang. “R-transformer: Recurrent neural network enhanced transformer”. In: *arXiv preprint arXiv:1907.05572* (2019).

[61] Yanjie Wen, Ping Xu, Zhihong Li, Wangtu Xu, and Xiaoyu Wang. “RPConvformer: A novel Transformer-based deep neural networks for traffic flow prediction”. In: *Expert Systems with Applications* 218 (2023), p. 119587. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2023.119587. URL: https://www.sciencedirect.com/science/article/pii/S095741742300088X.

[62] Xiping Wu, Hongyu Yang, Hu Chen, Qinzhi Hu, and Haoliang Hu. “Long-term 4D trajectory prediction using generative adversarial networks”. In: *Transportation Research Part C: Emerging Technologies* 136 (2022), p. 103554. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2022.103554. URL: https://www.sciencedirect.com/science/article/pii/S0968090X22000031.

[63] You Wu, Hongyi Yu, Jianping Du, Bo Liu, and Wanting Yu. “An Aircraft Trajectory Prediction Method Based on Trajectory Clustering and a Spatiotemporal Feature Network”. In: *Electronics* 11 (Oct. 2022), p. 3453. DOI: 10.3390/electronics11213453.

[64] Hua Xie, Minghua Zhang, Jiaming Ge, Xinfang Dong, and Haiyan Chen. “Learning air traffic as images: A deep convolutional neural network for airspace operation complexity evaluation”. In: *Complexity* 2021 (2021), pp. 1–16.

[65] Yan Xu, Xavier Prats, and Daniel Delahaye. “Synchronised demand-capacity balancing in collaborative air traffic flow management”. In: *Transportation Research Part C: Emerging Technologies* 114 (2020), pp. 359–376. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2020.02.007. URL: https://www.sciencedirect.com/science/article/pii/S0968090X19311295.

[66] Rikiya Yamashita, Mizuho Nishio, Richard Do, and Kaori Togashi. “Convolutional neural networks: an overview and application in radiology”. In: *Insights into Imaging* 9 (June 2018). DOI: 10.1007/s13244-018-0639-9.

[67] Junfeng Zhang, Jie Liu, Rong Hu, and Haibo Zhu. “Online four dimensional trajectory prediction method based on aircraft intent updating”. In: *Aerospace Science and Technology* 77 (2018), pp. 774–787. ISSN: 1270-9638. DOI: https://doi.org/10.1016/j.ast.2018.03.037. URL: https://www.sciencedirect.com/science/article/pii/S1270963816313414.

[68] Shuang Zhang, Rui Fan, Yuti Liu, Shuang Chen, Qiao Liu, and Wanwen Zeng. “Applications of transformer-based language models in bioinformatics: a survey”. In: *Bioinformatics Advances* 3.1 (Jan. 2023). vbad001. ISSN: 2635-0041. DOI: 10.1093/bioadv/vbad001. eprint: https://academic.oup.com/bioinformaticsadvances/article-pdf/3/1/vbad001/49324476/vbad001.pdf. URL: https://doi.org/10.1093/bioadv/vbad001.