Czech Technical University in Prague

Faculty of Transportation Sciences

Bachelor’s Thesis

Prediction of approach time to significant points in aircraft trajectories

Anu Bataa

Study Program: Technology in Transportation and Telecommunications

Study Field: Intelligent Transportation Systems

Thesis supervisors: Doc. Ing. Vit Fabera, Ing. Peter Haering

Prague 2022

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta dopravní
děkan
Konviktská 20, 110 00 Praha 1

K614..Ústav aplikované informatiky v dopravě

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení studenta (včetně titulů):

Anu Bataa

Studijní program (obor/specializace) studenta:

bakalářský – ITS – Inteligentní dopravní systémy

Název tématu (česky): **Predikce dosažení význačných bodů v trajektorii letu**

Název tématu (anglicky): The Prediction of Approach Time to Significant Points in Aircraft trajectories

Zásady pro vypracování

Při zpracování bakalářské práce se řiďte následujícími pokyny:

- Prostuduje distribuci letových dat poskytovaných platformou B2B Eurocontrol
- Proveďte rešerši již aplikovaných metod predikce na oblast význačných bodů a zhodnoťte je
- Seznamte se s používanými predikčními modely
- Navrhněte vhodný další predikční model pro predikci význačných bodů

ČVUT
FD

Rozsah grafických prací:	dle pokynů vedoucího práce
Rozsah průvodní zprávy:	minimálně 35 stran textu (včetně obrázků, grafů a tabulek, které jsou součástí průvodní zprávy)
Seznam odborné literatury:	Ivan Nagy: Základu Bayesovského odhadování a řízení, 2003 Mařík a kol.: Umělá inteligence 1 - 3 Ludvik Kulčák: Air Traffic Management, 2002

Vedoucí bakalářské práce: **doc. Ing. Vít Fábera, Ph.D.**
Ing. Petr Haering

Datum zadání bakalářské práce: **8. října 2021**
(datum prvního zadání této práce, které musí být nejpozději 10 měsíců před datem prvního předpokládaného odevzdání této práce vyplývajícího ze standardní doby studia)

Datum odevzdání bakalářské práce: **8. srpna 2022**
a) datum prvního předpokládaného odevzdání práce vyplývající ze standardní doby studia a z doporučeného časového plánu studia
b) v případě odkladu odevzdání práce následující datum odevzdání práce vyplývající z doporučeného časového plánu studia

Fábera Vít

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta dopravní
L. S.
- 2 -

[illegible]

doc. Ing. Vít Fábera, Ph.D.
vedoucí
Ústavu aplikované informatiky v dopravě

doc. Ing. Pavel Hrubeš, Ph.D.
děkan fakulty

Potvrzuji převzetí zadání bakalářské práce.

[illegible]

Anu Bataa
jméno a podpis studenta

V Praze dne 08.10.2021

Declaration

"I have no serious reason against the use of this work in the sense of § 60 of Act No. 121/2000 Coll., on copyright, on rights related to copyright and on the amendment of certain laws (Copyright Act)."

"I declare that I prepared the submitted thesis independently and that I have listed all the information sources used in accordance with the Methodological Instruction on the observance of ethical principles in the preparation of university final theses."

05.08.2022

[illegible]

In Prague

Anu Bataa

Acknowledgement

I would like to extend my deepest gratitude to my thesis supervisors Doc. Ing. Vít Fábera and Ing. Peter Haering for outstanding support and mentorship during the realization of this work. Also, special thanks should go to company CS-Soft a.s. for help and provision of accesses to required resources and data.
Lastly, I would like to thank my family and friends for being a part of this journey.

Contents

List of Figures

List of Tables

1. Abstract

This bachelor’s thesis deals with analysis of applications of probabilistic statistical and neural network models on spatial data for predictions of time in which the aircraft reaches the specified significant points in its trajectory and proposes an alternative method of prediction for temporal data.

Theoretical part of the thesis at hand is comprised of qualitative analysis of the existing methods applied to aircraft trajectory prediction on spatial data, definition of the key parameters, evaluation of the performance of the models. Theoretical definition of alternative method for temporal data obtained from flight planning, data extraction method, outline of its structure and its processing will be described in this part as well.

The practical part of the thesis describes the implementation of LSTM network on the temporal data and performs fine-tuning experiments for testing the possible usage and influence of a non-numerical parameter on the model performance.

Although, from the results obtained from the analysis, significance of geographical data for the approach time prediction was demonstrated, a time-series prediction made by an LSTM model exclusively using temporal parameters proved to produce viable results.

1. Abstrakt

Tato bakalářská práce se zabývá analýzou aplikací pravděpodobnostního statistického modelu a modelu neuronových sítí na prostorová data pro predikce času, ve kterém letadlo dosáhne zadaných významných bodů na své trajektorii, a navrhuje alternativní způsob predikce pro časová data.

Teoretická část práce se skládá z kvalitativní analýzy stávajících metod aplikovaných na predikci trajektorie letadla na prostorových datech, definice klíčových parametrů a vyhodnocení výkonnosti modelů. V této části bude také popsána teoretická definice alternativní metody pro časová data, metoda extrakce dat, nástin jejích struktury a jejích zpracování.

Praktická část práce popisuje implementaci LSTM sítě na časových datech a provádí *fine-tuning* experimenty pro testování možného použití a vlivu nenumerického parametru na výkon modelu.

Ačkoli z výsledků získaných analýzou byla prokázána významnost geografických dat pro predikci doby přiblížení, predikce časové řady provedená modelem LSTM výhradně s použitím časových parametrů prokázala proveditelné výsledky.

2. Introduction

Increasing air-traffic demand creates a strain on efficient operations of air-traffic control hence creating a growing interest and requirement for accurate prediction of approach times of individual aircrafts.
Accurate prediction of approach times enables the airports to prevent flight delays to a certain extent by mitigating high congestion levels at Terminal Maneuvering Area (TMA), to manage the air-traffic controllers' workload and to efficiently allocate ground resources for a consistent flow of operations.

There are several existing techniques of estimation and prediction of flight trajectories of specific aircrafts, such as statistical methods or prediction models based on artificial intelligence models. All the existing models are limited in their use due to the uncertainties caused by an array of factors surrounding the aircraft operations and environmental factors and require geographical and physical parameters as inputs to produce reliable results.

Trajectory prediction performed using location data obtained from radar entries can provide approach time estimates based on statistical or probabilistic neural network models implemented by extrapolating the predicted location from parameters such as the trajectory, geolocation, speed, altitude, and types of the aircraft.

The purpose of this work is to examine existing methods of prediction and describe an alternative method, in this case a time-series prediction model applied on temporal data, predicting actual times of arrival at a chosen significant point in aircraft trajectories. An application of an alternative method of prediction using time-series estimation is tested, in particular, to analyze the behavior of temporal data and evaluation of whether this use-case would be viable for such implementation through series of experiments examining the performance of a RNN model solely on temporal data and possibly tuning on other non-numerical parameters.

Outline of work. Section 4 is comprised of the theoretical part where two types of methods of prediction applied on spatial data obtained from flight radars are chosen and their performance and limitations are qualitatively assessed. Furthermore, a recurrent neural network – Long Short-Term Memory method is described along with the other component used in the practical implementation of the model. Section 5 outlines the practical part of the thesis and explains the principle of extraction of input data, their structure, and the description of their processing for implementation of the proposed LSTM model. Furthermore, this section includes the explanation of the practical implementation of LSTM model applied on temporal data along with a qualitative analysis to verify its performance and limitations through a series of experiments. An outline of a set of fine-tuning experiments is also included in this section. Section 6 provides the conclusion of the findings obtained from the analysis and experiments performed in the thesis.

3. Goals of the bachelor’s thesis

1) Analysis of prediction model applications on spatial and temporal data
2) Establishing and explaining a data extraction and processing method
3) Implementation and analysis of an LSTM model on a time-series data

4. Theoretical part

4.1 Air Traffic Control

European Organization for the Safety of Air Navigation (EUROCONTROL) is an international organization operating to achieve safe and seamless air traffic management practices across Europe. Their main operations include development and operations of air traffic management in various countries in Europe.[1]

The purpose of air traffic control is to ensure operational safety of air traffic and efficient utilization of ground resources. To attain these goals the agency utilizes prediction of aircraft approach times and prediction of trajectories.

Prediction of approach times of aircrafts to significant points is commonly done on location data and implemented using regressors such as geographical coordinates, altitude, speed of the aircraft, type, and request time. In this section we will be analyzing the principles of their implementation and evaluation of their results and describing a possible application of a Long Short-Term Memory Recurrent Neural Network model on the temporal data of aircraft arrival times and will include a description and analysis of it.

4.2 Probabilistic prediction analysis

In this section we will be analyzing the implementation and performance along with input data parameters from a bachelor's thesis by M.Kittler 2012 [2].

4.2.1 Probabilistic prediction model

Statistical models base their outputs on analysis of the correlation between specific variables to predict the outcomes that can occur in the future based on historical data. These types of models require a definitive understanding of parameters used as regressors, and the factors affecting them. Furthermore, these types of models require an extensive set of data for extrapolation of predictions. Due to these characteristics probabilistic models require a high computational performance and complexity in data pre-processing and analysis.

For the implementation of a probabilistic prediction model, a deeper understanding and interpretation of data is required. The analyzed model uses the parameters listed below for the estimation of prediction:

4.2.2 Probabilistic model implementation steps

The first step in implementation of the model is the initial understanding of flight operations and the factors affecting the data distribution. The outcome of the prediction depends on an array of physical factors that determine the trends in the development of the predicted value. Parametric probability density estimation is done by taking a large number of time intervals during which the planes get from one point to another, and it can be used to create a histogram where the interval with the highest frequency is then subtracted as the most probable estimate, which is taken as the final prediction result.

For the implementation of the analyzed model, input data consisting of radar records consisting of about 361 entries for each flight were obtained and restructured into a variables structure called “tracker”. The structures and parameters used for further processing are defined in a script which is called in every function to localize these variables in every function. The data is then filtered and grouped by separating radar entries for each flight and each entry is enriched with values such as time and remaining distance to and the altitude at the destination point. For improvement of accuracy the data is then cleared of mistakes in entries such as merging of identification information of separate flights due to use of same aircrafts with same identification variables for regular flights. These errors in entries are filtered with a specific script created for this purpose.

After the filtration of the input data, the prediction is made using the compulsory inputs below:

- Radar records
- Selected point for which the approach time is predicted

The prediction principle in based on creating a group of radar records which are similar to the one for which the prediction is made, and the most probable estimate is computed from the probability distribution of known approach times. The algorithm takes the input parameters for which it will be calculating a prediction and puts together similar historical records, and based on their distribution, chooses the most probable value, which is taken as the prediction result.

To calculate the prediction, in the analyzed work, the external factors potentially affecting the development of approach times have been grouped and studied for the determination of usable parameters for the next steps of the implementation.

4.2.3 Probabilistic model performance evaluation

In the evaluated experiment, the prediction is made based on probability distribution model of radar entries which were collected in the span of 14 days between the period of September and October of 2010, and the testing is performed on radar data obtained from one day in February 2011.

In the experimental scenario, when the prediction is made using the whole set of data without classification based on parameters evolving in time, the probable times of arrivals are constant on any set of input data, thus it would mean that the probabilistic model does not have a merit in such case. Therefore, there is a need for distinguishing specific parameters which change in time, such as:

- position of the aircraft
- close vicinity of the aircraft
- distance of the aircraft to the selected destination
- flight direction
- flight level
- flight speed
- place of departure (ADEP)
- place of landing (ADES)
- type of aircraft
- type of engine
- location of the target area
- required flight level in the target area

These input data parameters choose similar historical entries and put them into a set and create a probability distribution model. The prediction is based on evaluating the distribution of the set and choosing the most probable value which in turn comprises the result. It can happen that when the parameters will put together an empty array, in which case the probability model fails or produces a random result. Once such occurrence happens, it is possible to limit the input parameters and do the probability prediction again. Different parameters affect the probability distribution in various ways, therefore merging the evaluation parameters can improve the performance of the model.

In the examined work, individual parameters influence the prediction of time in different ways. The best results in prediction of the arrival times to the significant points was attained for the parameter proximity. Next, the combination of parameters produces accurate prediction results.
The tests performed in the examined work show that the time prediction for individual classes, using the narrowed set of parameters, is more accurate compared to the overall prediction from the basic set using the mixture of random variables.

In the evaluation of obtained results from the array of different tests, the author concludes that it is noticeable that for overflights and mainly direct flights, the probabilistic prediction without parameters does not achieve better result than the current method used by Eurocontrol based on the calculation of time from the distance and speed of the aircraft.
In the comparison of results obtained from tests using different parameters such as proximity, speed, altitude and direction, we observe an improvement in the forecast compared to the current method. This fact is related to the necessity of aircraft vectoring to the direction of the

runway used for landing or to the direction of the track after take-off in the opposite direction.

In conclusion, probabilistic method of prediction requires a comprehensive understanding of parameters in input data and statistical tests evaluating the importance of each of them needs to be performed in order to filter out the non-significant parameters and base the prediction on the relevant ones. Generic models of prediction based on whole set of data can produce static prediction and perform poorly due to it taking the probability distribution of the whole set of data for prediction.

The probabilistic method of prediction can perform well if the environment factors are stable.

4.3 Probabilistic Neural Network analysis

In this section we will be looking at a Probabilistic Neural Network implementation by P.Hroššo 2012 [3].

4.3.1 Probabilistic Neural Network

Another method worth analyzing can be PNN (Probabilistic Neural Network) based on Parzen estimates [3].

Parzen Window density estimation function is a non-parametric density estimation technique, used to compute the value of likelihoods on a new training sample x from the derived density function f(x) and returning the density estimate of the given data sample.

PNN is a feed-forward neural network in which connection between nodes don’t form a cycle. It calculates the probability density function of set of data and is used for classification and pattern recognition tasks. PNN is designed to solve classification problems by applying the idea of conventional probability theory to construct a neural network for classification.

The PNN topology consists of 4 layers:

- Input layer: p neurons represent the input layer and distribute it to the subsequent one. where p - number of input features.
- Pattern layer: layer which represents each training vector by a hidden neuron that records the features of this vector. During inference, each neuron calculates the Euclidean distance between the input test vector and the training sample, then applies the radial basis kernel function. In this way, it encodes the PDF centered on each training sample or pattern.
- Summation Layer: computational layer which calculates the output of the pattern units for each class. This layer contains one neuron for each class which is then connected to all neurons in the pattern layer of that class.

- Output Layer: this layer finds the maximum value from the previous layer and determines the associated class label.

PNN can be a reliable classifier, however, it requires a smoothing parameter in case of a limited dataset for a better performance.

4.3.2 Probabilistic Neural Network implementation steps

The data used in the analyzed work is obtained from Air Traffic Management authority which included radar data entries along with added entries from flight plans. It consists of flight entries with recorded parameters below:

- Flight ID
- Call sign
- Type of aircraft
- Aircraft description (number of motors etc.)
- Aerodrome of departure
- Aerodrome of destination
- Radar entry recording time
- Location of the aircraft
- Altitude
- Speed

The data contained flights both from and to Prague and was additionally filtered out to contain only radar data entries which is then filtered again based on parameters since only some are used for training of the PNN. Lastly, it is important to ensure that the data entries have the same dimensions, and the temporal parameters are discreet for the PNN to be able to classify.

The training process consists of splitting the dataset into training and testing sets and feeding the training data directly to a PNN weight matrix.
The process of classifying the radar records is as follows: the author calculates the matrix of activations of all neurons as a matrix product of all the weights of our PNN and the radar input vector. For faster classification of a larger number of inputs the vectors were merged into one matrix of all inputs, which speeds up calculations.

For these activations, the value of the activation function was calculated. Then the author goes through all distinguished classes and sums the contributions of individual neurons. As a result, the classification is the class with their highest sum.

4.3.3 Probabilistic Neural Network performance evaluation

In the examined work, the author performed tests in which they verified the effects of training on the success rate of prediction. A satisfactory classification success rate was observed with the ratio of training and testing data set splitting with a ratio of one to one, where the success rate was approximately 85%. The success rate continued to improve with the increase of the training set up to 95%.

Next, the investigation of the effect of the number of distinct classes on classification success was done and resulted in a conclusion that the ideal number of classes for this use-case, is 150. In such a case, the prediction of time is made with of approximately 30 second accuracy, yields in high classification success rate (~94%) and the standard deviation is in seconds. The time performance of the prediction was analyzed and concluded that this figure depends on the technique used and an observation that the classification time increases linearly with the number of training data and classified sets was noted.

The model at hand was also compared with a naive linear estimator. For flights similar to flights in the training set, the predicted time was almost indistinguishable from the real time. With more complicated testing conditions, the model predicted a flight time that was significantly different from the flights in the training set.

Overall performance of the predictor was good, with the largest prediction error around 100 seconds. With regards to the influence of individual parameters of input data, highest weight in the prediction model is the position of the aircraft, followed by a group of inputs: flight level, speed, and type of aircraft. Inputs with lower weight are time of day and (takeoff/landing) runway. However, their influence cannot be neglected because they improve the prediction success by about 8% [3].

4.4 Recurrent Neural Network

In the practical part of this bachelor's thesis we will implement an LSTM model on our data. In this section, we will be looking at the theoretical description of models of this type and their general architecture.

4.4.1 Recurrent Neural Network on temporal data

Time series prediction models are a difficult type of predictive modelling due to the addition of complexity of sequence dependence in input variables. Such sequence dependencies are handled with a specific type of neural networks called recurrent neural networks.

Recurrent neural network (RNN) is a type of artificial neural network which uses sequential, time series data and similarly to feedforward convolutional neural networks, utilizes learning on training data. RNN is commonly used for problems such as language translation, natural language processing, speech recognition and image captioning. The distinguishing aspect of recurrent neural networks is the internal memory which maintains the dependence of outputs on prior elements within the sequence, compared to assumption of input and output independence in traditional deep neural networks. Recurrent neural networks need to account for the attributes and sequence of previous elements to produce a prediction output. Another distinguishing characteristic of the RNNs is that they possess an ability to propagate parameters across the layers of the network, meaning that the network shares the same weight parameter within each layer of network, which are then adjusted through the processes of backpropagation and gradient descent to facilitate reinforcement learning. [4]

RNN leverages backpropagation through time algorithm (BPTT) for determination of the gradients which is an approach specific for sequence data. BPTT sums errors at each time step to adjust the model parameters to fit appropriately. Due to this characteristic, RNN models tend to encounter two distinct problems: exploding and vanishing gradients. These issues are defined by the size of the gradient, which is the slope of the loss function along the error curve. When the gradient is small it continuously decreases, updating the weight parameter to approaching 0. In this situation the learning of the algorithm is stalled or is stopped. On the contrary, exploding gradients occur when the weights of the layers inflate and become unrepresentable.

4.4.2 LSTM on temporal data

Long Short-Term Memory model has been introduced in 1997 by Sepp Hochreiter and Juergen Schmidhuber as a solution to mitigate the occurrence of vanishing gradients.
The model achieves this by utilizing memory cells and gate units in its architecture that allows constant error flow through self-connected linear units – constant error carrousel (CEC). [4]

The model is based on an architecture built on memory cells (topology pictured in figure 1), which is a more complex unit, compared to neurons in traditional neural networks, built around a CEC and has three distinct gates – an input gate, an output gate, and a forget gate. A multiplicative input gate unit protects the memory contents of the memory cell from perturbation from irrelevant inputs and a multiplicative output gate unit is used to mitigate the perturbation produced by irrelevant memory of the current cell on other units.

The topology of the network consists of one input layer, one hidden layer and one output layer. The fully self-connected hidden layer contains memory cells with corresponding gate units. All units apart from the gate layer units have directed connections to all units in the layer above or higher, which serve as an input. Layers are comprised of S memory cells sharing the same input and output gates and are referred to as “memory cell block of size S”, which facilitate information storage.

The mathematical model of the LSTM RNN can be represented as below [5]:

$$y^{out_j}(t) = f_{out_j}(net_{out_j}(t)); y^{in_j}(t) = f_{in_j}(net_{in_j}(t));$$

[5]

where

$$net_{out_j}(t) = \sum_u w_{out_j u} y^u(t-1),$$

and

$$net_{in_j}(t) = \sum_u w_{in_j u} y^u(t-1).$$

The learning of the network uses a variant of Real Time Recurrent Learning, which is adapted considering the multiplicative nature of dynamics of input and output gates, hence mitigating the occurrence of the vanishing gradient problem. This is achieved by ensuring non-decaying error backpropagation through the internal states of memory cells, where the errors arriving at memory cell inputs do not get propagated further back but serve for adjustment of the incoming weights. In other words, once the error signal arrives at a memory cell output it gets scaled by output gate activation and differential of hidden state function h.

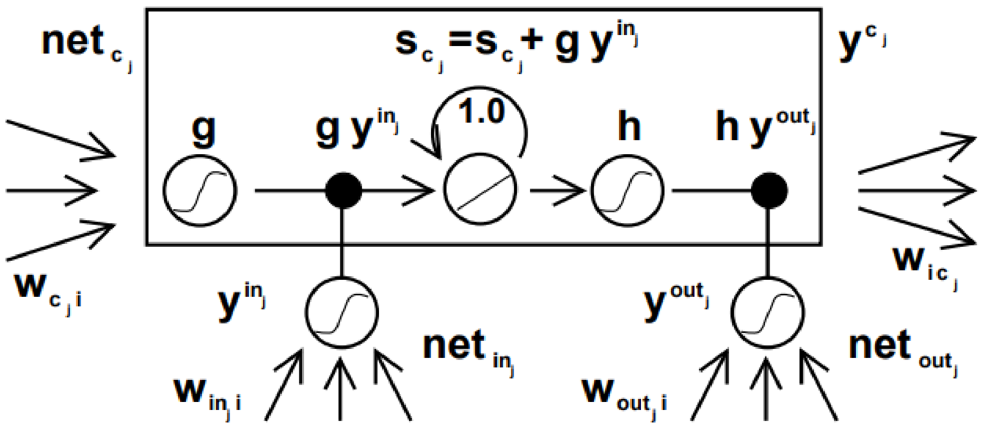

Fig.1 Architecture of a memory cell of a LSTM model [4]
Architecture of memory cel l cj (the box) and its gate units inj ; outj . The self-recurrent connection (with weight 1.0) indicates feedback with a delay of 1 time step. The gate units open and close access to CEC.

LSTM is an efficient model in regards of computational difficulty, as only the derivatives should be stored and updated. The model can have drifts in internal state in case of input values being mostly positive or mostly negative, which in turn, can cause the gradients to vanish. This is mitigated by biasing the input gate toward zero.

4.5 Other components used in the implementation of proposed LSTM model

Successful implementation of machine learning models involves choosing appropriate algorithms for data interpretation and optimization. For our model, we will be performing statistical and autocorrelation testing for data interpretation, as well as Adam optimizer algorithm for the LSTM model.

4.5.1 Autocorrelation in time-series data

Autocorrelation refers to the degree of similarity between given time series and its lagged version over a specific timeframe. One-step lag autocorrelation would be demonstrating whether the present time-series have any correlation with its lagged values. Autocorrelation can detect correlation pattern or trends in data and is used as a step in data analysis prior to choosing a suitable model for prediction. It plots the current values with lagged ones on a graph and calculates the degree of similarity between those. There can be as many degrees of lags as possible, depending on the number of entries in the input data, however the reliability of autocorrelation model decreases dramatically with each added step, as there are less entries available for each of the added lag [6].
It is important to test the data for autocorrelation as in some instances, the presence of autocorrelation would signal to RNN models to be not suitable for that set of data.

4.5.2 Optimizer Adam

Optimization algorithm is a procedure which is executed iteratively comparing different possible outputs to find the optimal or the satisfactory solution. Many fields of science and engineering require maximization or minimization of parametrized objective functions with respect to its parameters. For many objective stochastic functions derived from sum of subfunctions which are evaluated on different subsets of data, the optimization can be more efficiently made using stochastic gradient descent or ascent. For functions with more noisy objectives, a method with high-dimensional parameter space is more suitable.
Adam is an efficient algorithm for stochastic optimization which requires first-order gradients and does not need a big memory. The name is derived from "adaptive moment estimation" and is it built to combine advantages of AdaGrad (Duchi et al. 2011), for problems with sparse gradients, and RMSProp (Tieleman & Hinton, 2012) which is suitable for on-line and non-stationary settings. [7]
Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance).
The algorithm calculates an exponential moving average of the gradient and the squared gradient, and the parameters beta1 and beta2 control the decay rates of these moving averages.
The initial value of the moving averages and beta1 and beta2 values close to 1.0 result in a bias of moment estimates towards zero. This bias is overcome by first calculating the biased estimates before then calculating bias-corrected estimates.
Adam utilizes the configuration parameters below [7]:

- alpha. Also referred to as the learning rate or step size. The proportion that weights are updated (e.g. 0.001). Larger values (e.g. 0.3) results in faster initial learning before the rate is updated. Smaller values (e.g. 1.0E-5) slow learning right down during training
- beta1. The exponential decay rate for the first moment estimates (e.g. 0.9).

- beta2. The exponential decay rate for the second-moment estimates (e.g. 0.999). This value should be set close to 1.0 on problems with a sparse gradient (e.g. NLP and computer vision problems).
- epsilon. Is a very small number to prevent any division by zero in the implementation (e.g. 10E-8).

Adam is the recommended optimizer for deep learning algorithms.

5. Practical part

5.1 Dataset obtained from EUROCONTROL

The data used in the implementation of the proposed LSTM model is acquired from Eurocontrol B2B database which is accessed through the B2BD – CS SOFT Gateway interface ensuring certified access. B2BD mediates B2B data to other clients through Redis database.

5.1.1 Dataset structure

We are using a dataset containing a collection of entries of registered flights arriving to Vaclav Havel Airport (ICAO code: LKPR) over the span of three days, collected in intervals of 5 minutes. Each request obtains a result in the defined format which is then extracted by a script into a CSV file. The data consists of 347 different flights originating from 121 aerodromes and records below values for each registered flight:

- sendTime - timestamp of the request time
- requestID – ID of the data request
- effectiveTrafficWindow_wef – timestamp of the data snapshot
- effectiveTrafficWindow_unt – timestamp of the data snapshot
- flightId – ID of the individual flights
- aircraftId – ID of the aircraft
- aircraftType – model of the aircraft
- aerodromeOfDeparture – ICAO code of the departure aerodrome where the flight originated from
- estimatedOffBlockTime – estimated time when the aircraft vacates the parking slot
- estimatedTimeofArrival – time of arrival in the flight plan
- actualTimeOfArrival – time of arrival predicted in real time by EUROCONTROL
- flightState – state of the flight

5.1.2 Principle of data extraction

CSS-B2BD created by CS-Soft a.s. is a server daemon that, on one hand, ensures certified communication with Eurocontrol B2B and, on the other hand, mediates this communication to clients via a Redis database API. Python scripts access Eurocontrol B2B exclusively via CSS-B2BD, i.e exclusively via Redis API.

The Redis API is built on pairs of Redis communication channels through which requests and responses are transmitted. The query is sent through the Request channel when the request script performs the “Publish” operation. The response will then come through the Response channel once the script waits with the “Subscribe” operation.

The API uses the naming of communication channels according to the following format: `‘css:b2b:req|rep:<ident>:<num>’`.

Query and Response are data structures that contain requested parameters and structured response data. CSS-B2BD supports two formats representing a different level of Eurocontrol B2B data processing:

- RAW-XML: queries and response data are in XML format, which directly corresponds to the Eurocontrol B2B specification. CSS-B2BD only mediates communication with Eurocontrol B2B.
- JSON: CSS-B2BD simplifies common queries and responses into JSON structures to save and simplify repetitive work for operational applications. The structure of both the request and the response are more understandable in this format.

For this work, the RAW-XML format was used because it is generic and valid even in the absence of CSS-B2BD, as it has all the available features provided by Eurocontrol B2B. For operational systems, JSON may be more advantageous, where CSS-B2BD can be prepared to provide applications with only necessary data entries and parameters. The request can be structured to ask for any of the available parameters for each flight and in our case, the request asks for the parameters listed in section 5.1.1. The difference between the individual formats can be seen in request and response examples in Appendix 2.

5.2 Data interpretation and pre-processing

To ensure the functionality of prediction model the input data should be interpreted and pre-processed. The proposed LSTM model is concerned with prediction based on temporal data, therefore a suitable re-formatting and filtering is required to enable the application of time-series. The LSTM model itself can recognize and capture patterns in temporal parameters and train on them. However, apart from numerical values, the data entries also possess non-numerical parameters, such as aerodrome of departure and type of aircraft, which can hold significance in the training. Therefore, a further analysis of these parameters is required to establish their significance in the training model.

To use the raw input data in subsequent steps, we have merged the individually extracted CSV files into one dataset and have filtered out empty and duplicate values using pandas and numpy packages in Python, implementations of statistical tests are provided by scipy. (The script for formatting data: dataset.py)

5.2.1 Evaluation of parameter aerodrome of departure for subsequent fine-tuning

One of the parameters of interest is the aerodrome of departure, particularly, the direction of the originating aerodrome in relation to LKPR for which we are making the prediction of the delay in times of arrival. We will be testing a hypothesis that this parameter holds significance in estimating the delay of the aircraft. If the hypothesis holds true, using this parameter for fine-tuning the LSTM neural network would produce a better fitted model of prediction, or on the contrary, the rejection of this hypothesis would indicate that the generic model and fine-tuned models would not have a significant difference in accuracy.

At the start of this processing, the aerodromes occurring in our dataset were grouped into 4 groups based on their relative direction to the LKPR: *North*, *South*, *West*, and *East*.
Next, for each direction, the median time difference between estimated time of arrival and actual time of arrival among all entries was extracted. The obtained results are displayed in figure 2.

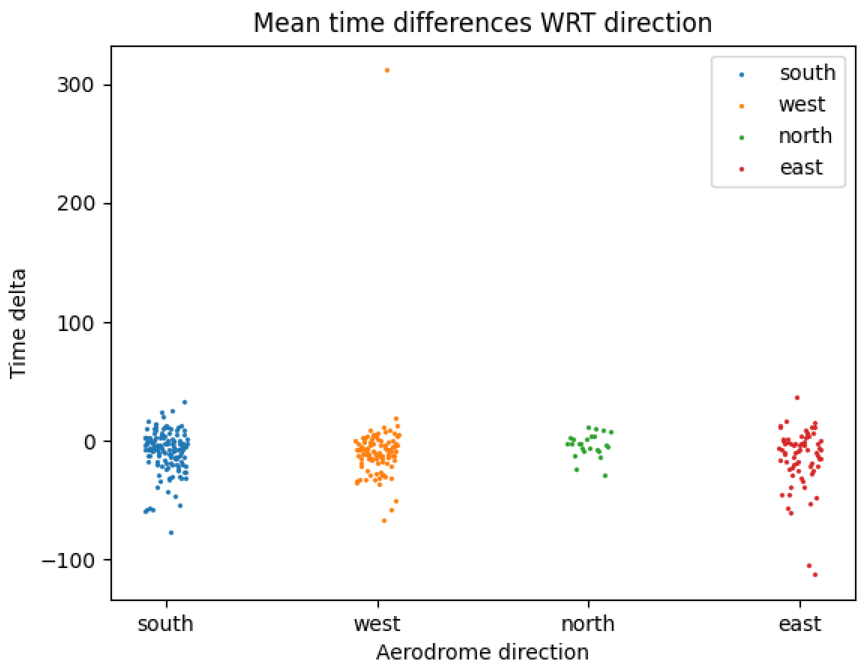

Fig2. Mean time difference between estimated time of arrival and actual time of arrival grouped by direction of Aerodrome of departure relative to LKPR

To determine significance of the flight direction as a training parameter, we need to perform statistical tests.

For this purpose, one-way ANOVA test was initially chosen, and it requires the below preconditions to be met [8]:

1. Data sets in each group must be independent
2. Each sample must be from a normally distributed population
3. The population standard deviations of the groups are all equal

For the first precondition of independence, we can conclude that the samples are independent, as each flight can originate from exactly one aerodrome at a time.

For the precondition of normality, we will perform Shapiro test on each of the sample groups with the given null hypothesis [9]:

H_0: “The sample data was drawn from a normally distributed population.”

We obtain the following values displayed in table 1 for the significance level of $\alpha = 0.05$:

	Statistic	**p-value**
South	0.935	5.48418e-06
West	0.403	2.65746e-18
North	0.946	0.22966
East	0.866	2.33677e-06

Table 1. Results of ANOVA test performed on datasets grouped by direction of Aerodrome of departure relative to LKPR

$p < \alpha$ holds in three out of the four cases, thus, we reject the null hypothesis.

As the precondition of normality was not met in this case, likely due to the small size of sample data, we cannot reliably use ANOVA test to determine the significance of this parameter.

Therefore, as a second option, we will perform Kruskal-Wallis test, which does not require the samples to be normally distributed. [10]
We propose the following null and alternative hypotheses for Kruskal-Wallis test:

H_0: “The directions of departure aerodromes cause the same delay. The samples have been drawn from the same or identical population.”
H_A: “At least one direction of departure aerodromes causes a different delay. At least one group is drawn from a different population.”

Applying the test on our data produced the following results:

$$T = 7.0851$$
$$p = 0.0692$$

Given that $p > \alpha$, the null hypothesis is not rejected. However, the values of p and α are relatively close. These findings can be caused by the limited size of the sample data and the proximity of values between $\alpha = 0.05$ and p can be signaling to existence of other requirements for better utilization of this parameter. Empirical experiment can be used to further elaborate this finding. The results of the Kruskal-Wallis test does not reject the hypothesis that the medians of the samples in the same. However, the samples may possess other parameters which can have a differing behavior, which can be picked up by the LSTM network.

5.2.2 Time series analysis

In this section, we take a closer look at the development of the difference between the estimated and the actual time of arrival of the aircrafts for individual flights. An example of such development can be seen in the figure 3, which has a visualization of the development of the difference between parameters *estimatedTimeOfArrival* and *actualTimeOfArrival* (Time delta) for specific flights. The data is filtered through the individual flight IDs and time series are created using the request times. Figure 3 demonstrates the development of "time delta" in time-series for 10 separate flights over the span of one hour.

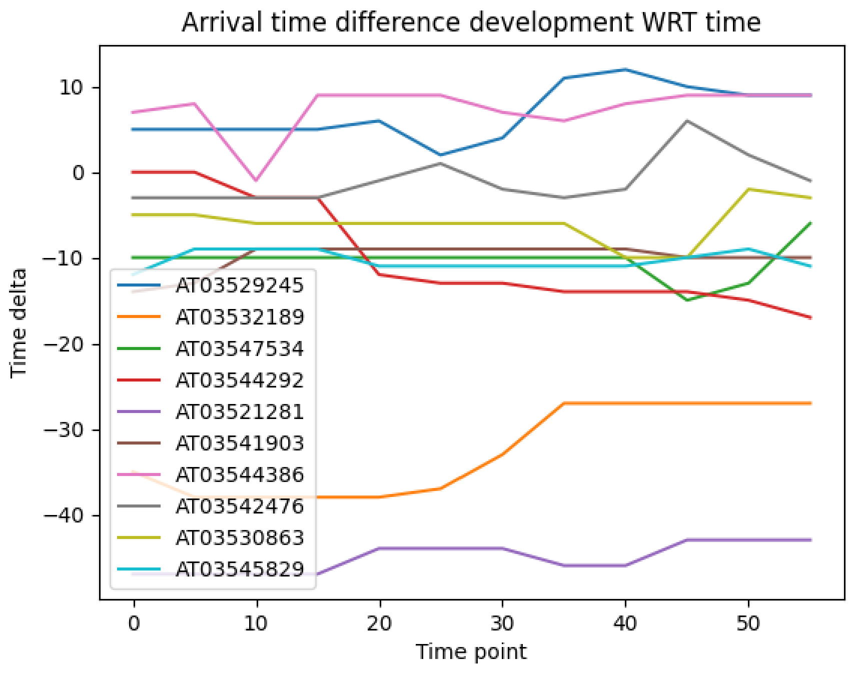

Fig. 3 Time-series plot of 10 flights with the greatest number of entries by the flight ID, collected in the span on 1 hour with 5 minute intervals. x-axis: time points in minutes, y-axis: subtracted difference between estimatedTimeOfArrival and actualTimeOfArrival

From the above visualization, the recognition of patterns is inconclusive as some flights have a more stable *actualTimeOfArrival* development in time and some have a higher degree of variation.

To better understand the data and to rule out the possibility of using mathematical extrapolation for the prediction on the input data, we have performed an autocorrelation test with one-step time lag.

Autocorrelation with a step of size 1 (in intervals of 5 minutes) of the time deltas for 10 randomly selected flights are presented in table 2 below:

Flight	AT035 29245	AT035 32189	AT035 47534	AT035 44292	AT035 21281	AT035 41903	AT035 44386	AT035 42476	AT035 30863	AT035 45829
ρ	0.614	0.833	0.004	0.722	0.544	0.468	-0.094	0.333	0.224	0.122

Table 2. Results obtained from autocorrelation test with 1 step lag. ***ρ*** *– autocorrelation, ranges between the values [-1,1]*

From the results obtained from autocorrelation, we can conclude that the predictions cannot be based on mathematical extrapolation, as the corresponding values with one-step time lag are producing a varying range of results for each flight and therefore can be deemed inconclusive. The flights with positive extrapolation suggest that there can be a specific pattern, however, the inconsistency suggests that a more sophisticated model of prediction is required for this estimation. To capture any trends and patterns in this time series and to predict the next value in them, we propose a LSTM neural network. [11]

5.3 LSTM

LSTM model performs well on time-series data, therefore we wanted to evaluate the performance of this model on our data for approach time prediction, in a scenario where geographical data is not taken into account.

5.3.1 LSTM Implementation

In the following experiments, we are considering a *LSTM* model with 4 input neurons, for the implementation of which, each flight needs to consist of at least 4 entries to be considered relevant. Thus, any flights with less than 4 entries are filtered out.
However, there is also a possibility of tuning this parameter out in the network configuration.

For the prediction of differences in estimated times of arrival and actual times of arrival, we are using a LSTM model consisting of two hidden layers with 64 neurons, or more precisely, memory cells. This configuration was chosen after considering similar implementations, and then tuning for our use-case, as the implementation of these models is commonly based on experimental approach. [12]

For our experiment, we are using a time series of four entries as the input for the network. For different set of data this parameter can be easily configured, as the model can be trained on inputs of different lengths.

The model uses ADAM optimizer and *mean-squared-error* as the loss function.
For the training process and subsequent testing we have split out dataset into training and testing batches, with a ratio of 9:1 (*90%* of the data from data set is used for training and remaining 10% for testing)

The implementation of the model is done in python, utilizing the packages listed below:

- Tensorflow
- Keras
- Sklearn
- Numpy
- Pandas

The full script can be found in the appended file 'lstm.py'.

The lstm.py script creates a dataset matrix and restructures the data to fit the LSTM model. The model itself is taken as a pre-defined algorithm, in our case a function which takes the training data as an input and based on the patterns recurring in the set, tunes the internal weights of memory cells, semantic representation of which is taken as a "black-box", to fit the data.

The training process of our model uses 90% of the dataset and the data is split for this purpose randomly. The ratio of 90 to 10 is chosen for the purpose of increasing the volume of training data.

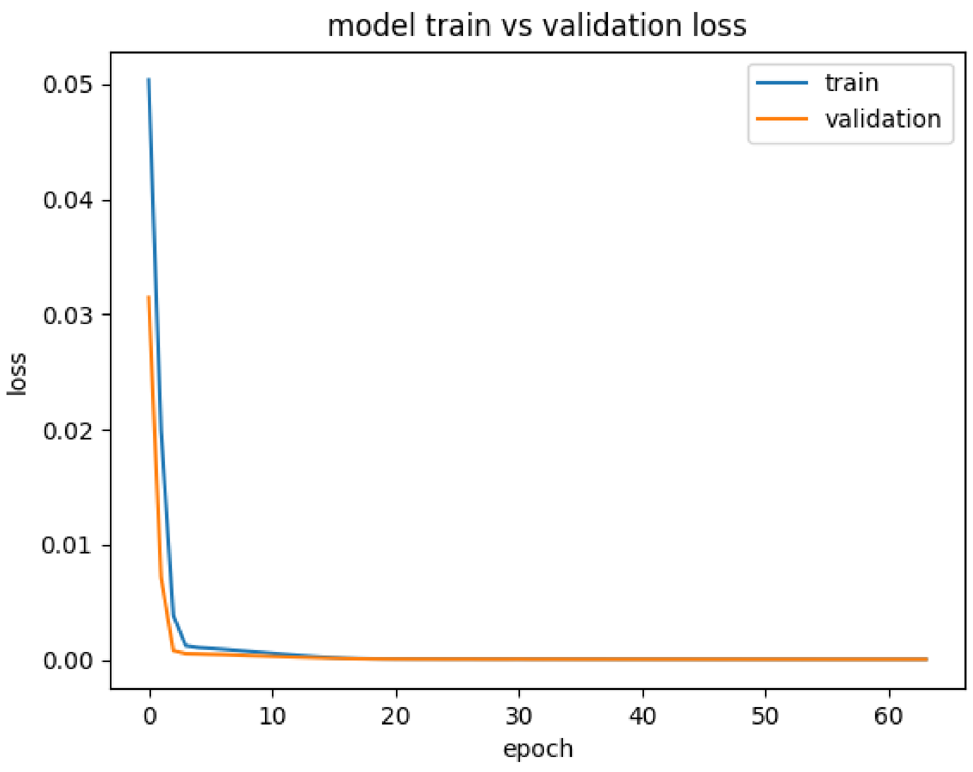

Fig. 4 Development of training vs. validation loss of the LSTM model over training epochs

From Figure 4, we can observe that the loss function converges at around 16 epochs of training. The development of loss function over the training epochs demonstrates the change in errors after consecutive training epoch. By the end of the training, there is a slight increase in the validation loss, which usually occurs when the model starts getting over-fitted. In our case the increase is minimal, ensuring that the model training balance is still attained.

The loss function values differ from the RMSE, due to the normalization of data for the network input.

5.3.2 LSTM generic model performance

Figure 5 demonstrates the prediction results on training data set.
The ground-truth values (blue) of the training set were sorted in an ascending order and are plotted together with the predictions (orange).

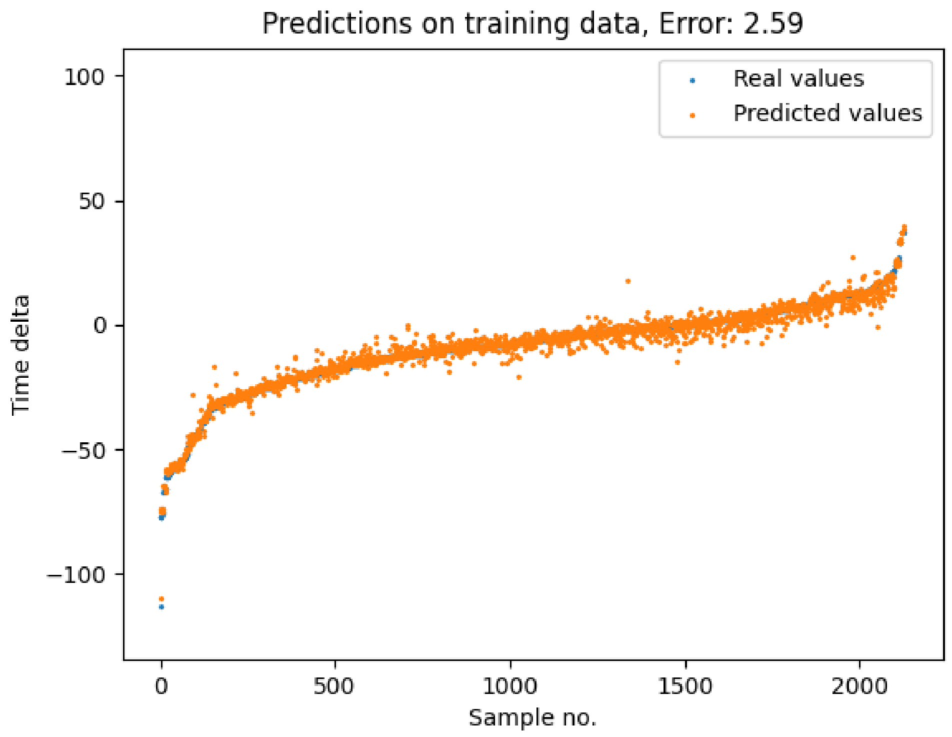

Fig.5 Plot of predicted and real values on training data

Once the training is completed, we can test the performance of the model on the testing set of data. The predicted values are displayed in figure 6. The prediction on testing data has a mean squared error of 2.46 minutes.

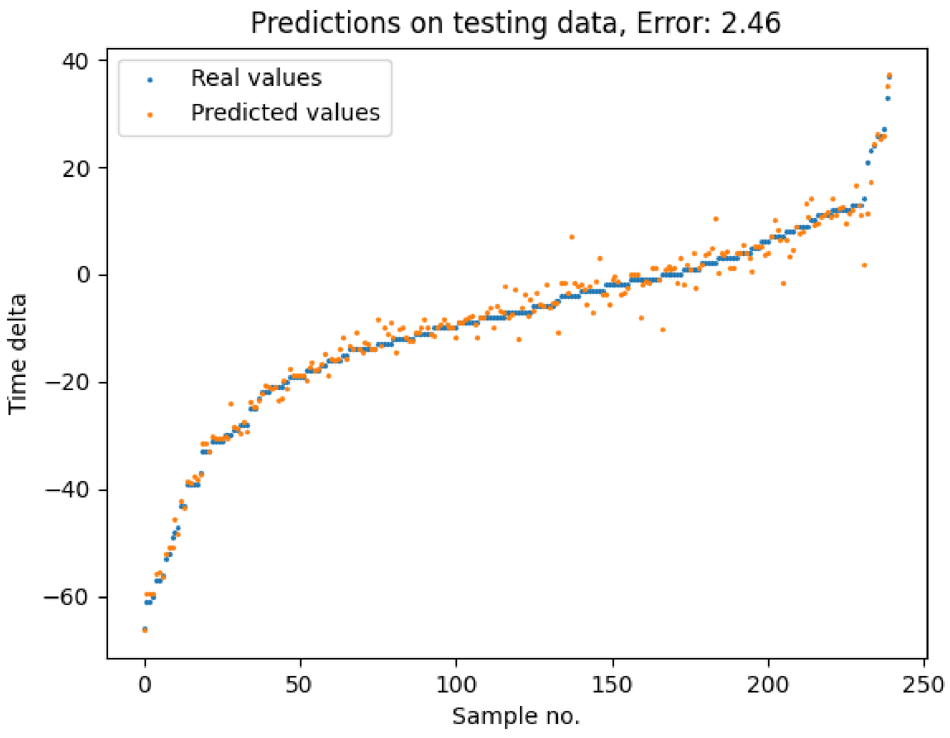

Fig. 6 Plot of predicted and real values on testing set of data

5.4 Fine-tuning experiment

In chapter 5.2.1 we did not reject the hypothesis, that entries grouped by the direction of the aerodrome belong to the same population. However, to elaborate further on this matter, we can propose the following experiment.
We will be fine-tuning the trained model specifically on data extracted from flights incoming from each specific direction for 4 epochs. After that, we will be testing the error of this newly trained model against the input data.

The following set of tables contains values observed during 8 different trainings.

South	1	2	3	4	5	6	7	8
Generic	2.49	2.70	2.95	2.30	2.54	2.50	2.63	2.81
Fine-tuned	2.46	2.68	2.95	2.27	2.53	2.49	2.71	2.92

Table 3. RMSE values of generic vs. South fine-tuned models from 8 consecutive experiments

East	1	2	3	4	5	6	7	8
Generic	1.63	2.03	2.24	1.67	1.95	2.10	1.53	2.04
Fine-tuned	1.60	2.02	2.14	1.57	1.97	2.04	1.43	2.06

Table 4. RMSE values of generic vs. East fine-tuned models from 8 consecutive experiments

North	1	2	3	4	5	6	7	8
Generic	2.05	1.87	2.14	2.52	2.52	3.58	2.00	2.09
Fine-tuned	2.04	1.85	2.22	2.56	2.51	3.35	2.00	2.10

Table 5. RMSE values of generic vs. North fine-tuned models from 8 consecutive experiments

West	1	2	3	4	5	6	7	8
Generic	2.96	1.40	2.74	3.12	2.83	2.98	2.83	2.63
Fine-tuned	2.90	1.42	2.68	3.06	2.78	2.98	2.82	2.74

Table 6. RMSE values of generic vs. West fine-tuned models from 8 consecutive experiments

Following figures visualize the different performance of the fine-tuned and generic models.

5.4.1 Fine-tuning by direction South

In case of the flights coming from the South to LKPR we can see a slightly better performance, with mean squared error in five out of eight experiments being 0.01 minutes less than the generic model. Generally, the difference of RMSEs of generic model and fine-tuned model on flights from the South do not produce a significant improvement. The below figure visually represents the results by plotting the observed values in comparison with predicted values of the generic model and fine-tuned model for direction South.

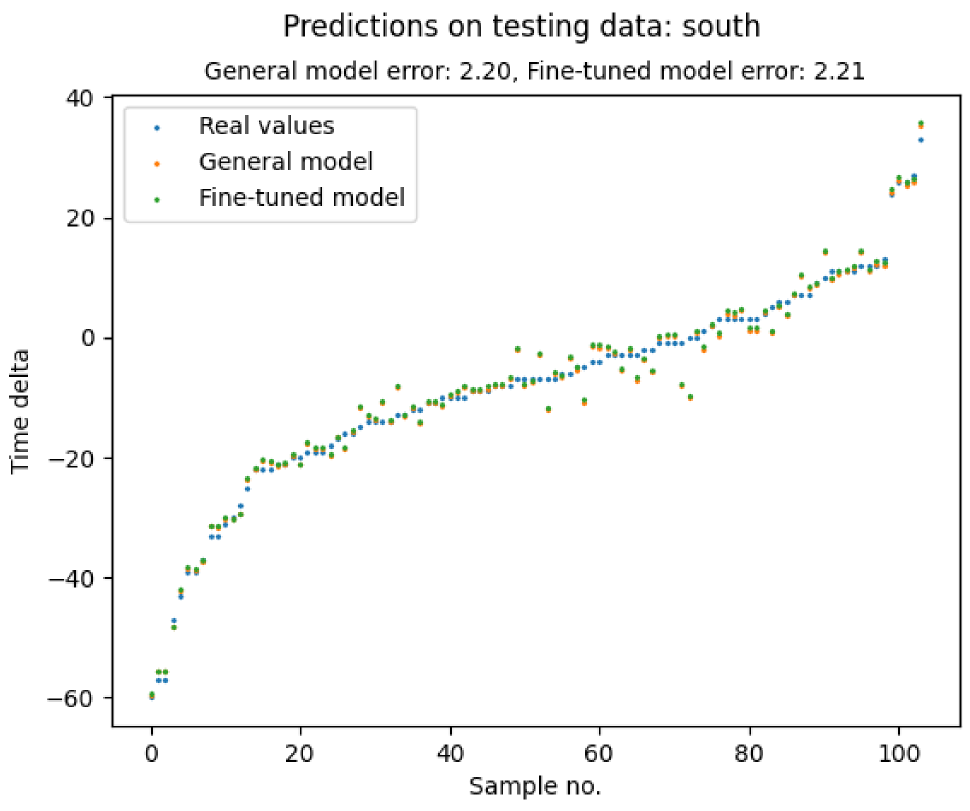

Fig. 7 Plot of Predicted vs Real values for the generic model and fine-tuned model for the departure aerodrome direction South, relative to LKPR

5.4.2 Fine-tuning by direction East

The performance of the fine-tuned model for direction East on overall produces similar results as for the South. In this case the RMSE is a bit less, therefore, we can conclude that the model fine-tuned on the flights from the East generally performed slightly better than the generic model on seven out of 8 experiments made for this direction.

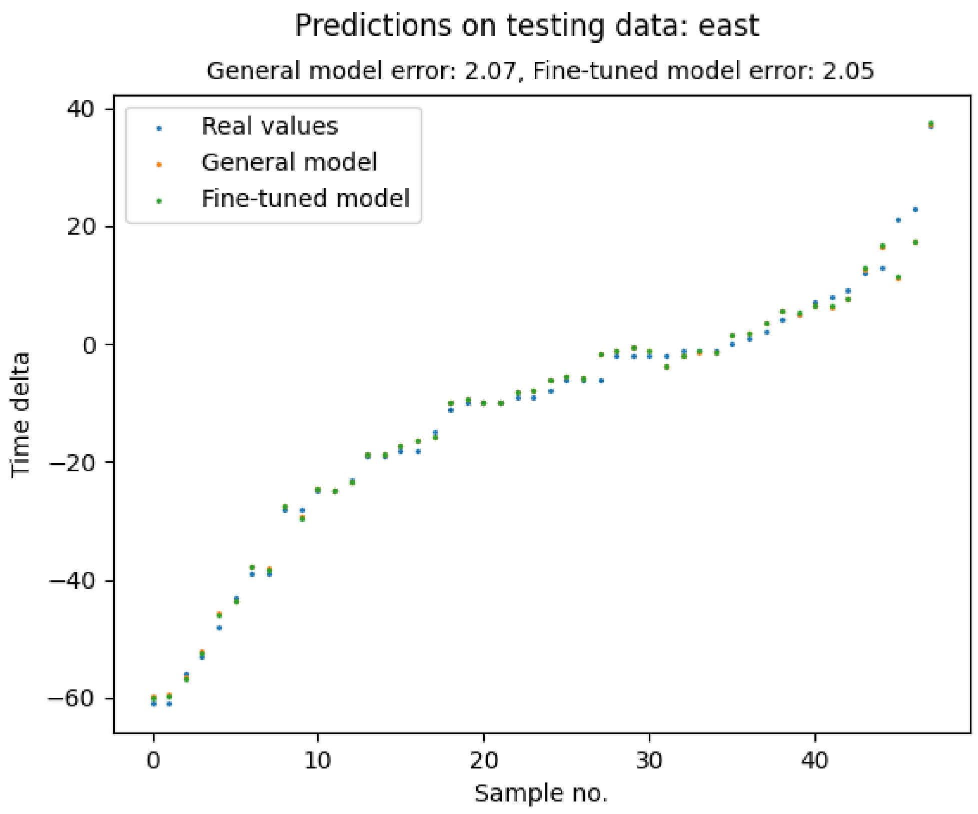

Fig. 8 Plot of Predicted vs Real values for the generic model and fine-tuned model for the departure aerodrome direction East, relative to LKPR

5.4.3 Fine-tuning by direction North

The model which was fine-tuned on the group of flight entries approaching LKPR from the geographical North, has performed in a similar manner as the previous models with five out of 8 experiments producing slightly better RMSE results. However, the difference is still very small to be considered significant.

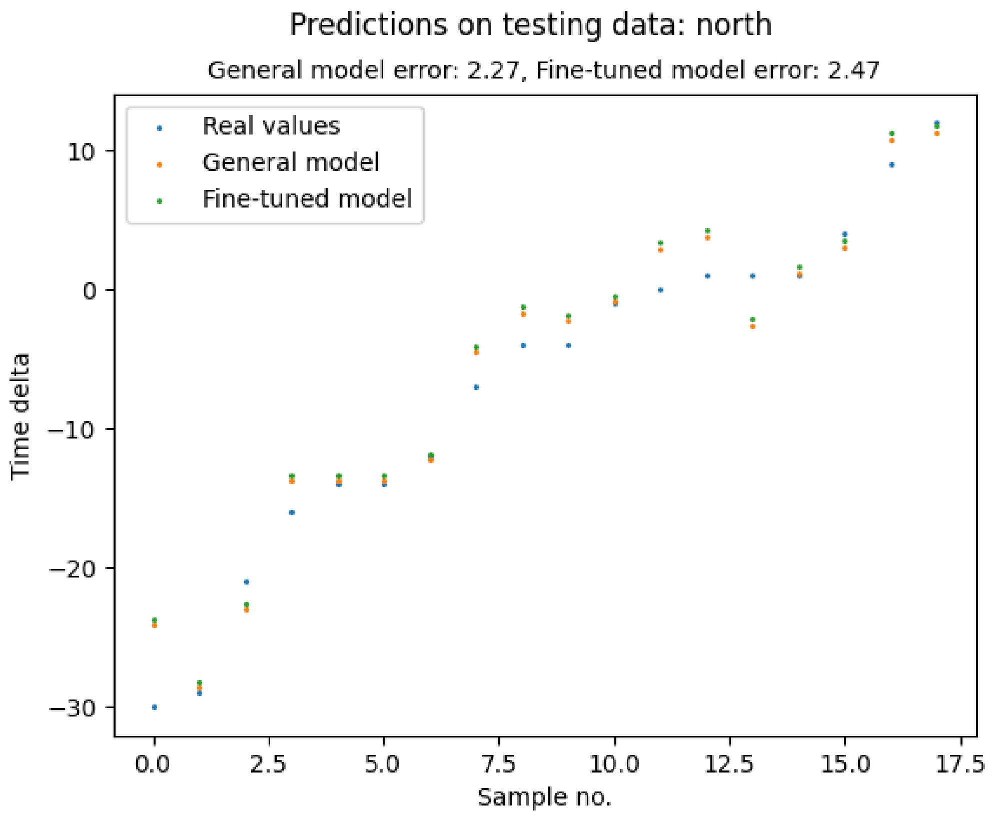

Fig. 9 Plot of Predicted vs Real values for the generic model and fine-tuned model for the departure aerodrome direction North, relative to LKPR

5.4.4 Fine-tuning experiment by direction West

The results obtained from the series of experiments on the entries of flights from geographical West produce a similar result to the rest of the directions with the performance being on average slightly better than the generic model on five out of eight instances.
The difference in values is still insignificant for the model to be considered as performing better than the generic one.

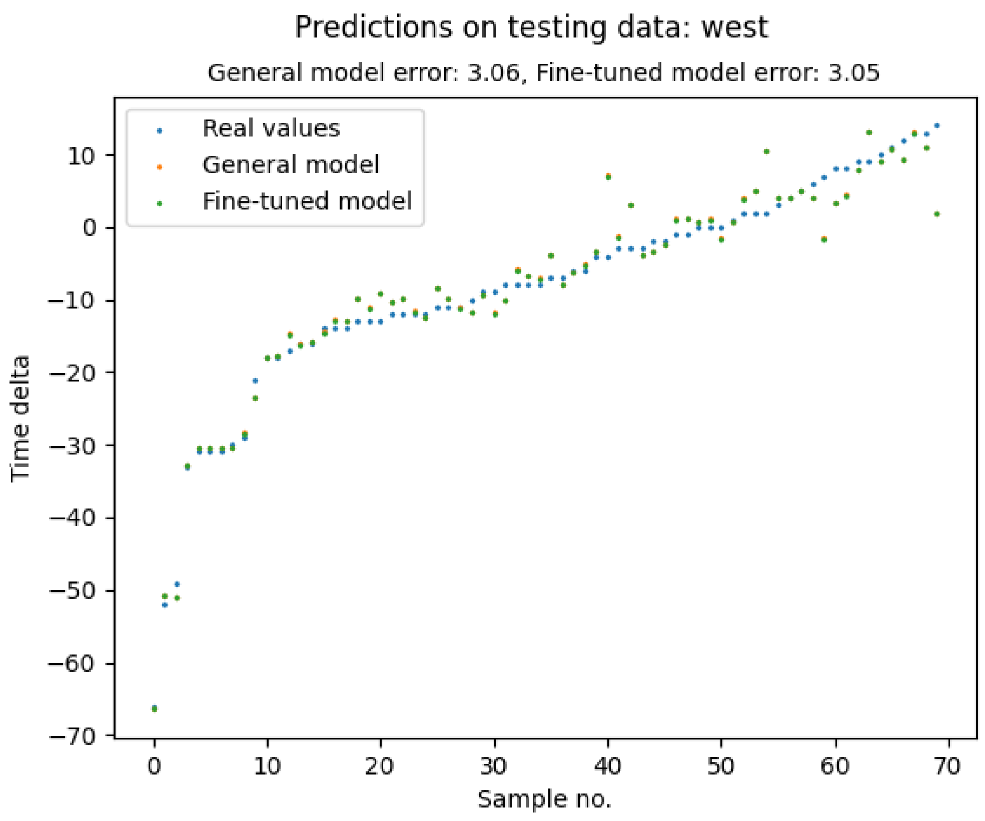

Fig. 10 Plot of Predicted vs Real values for the generic model and fine-tuned model for the departure aerodrome direction West, relative to LKPR

5.4.5 Fine-tuned model performance

From the results obtained from the comparison of the performance of the generic model against fine-tuned one we can see that for the use-case at hand, tuning does not produce a significant improvement for the LSTM model. This can be due to the grouping of the dataset into relative directions of aerodromes of departure to LKPR being too generic and a more detailed grouping of data entries could be producing a better parameter for fine-tuning. One of parameters for possible future experiments could be the distance of the aerodrome of departure from LKPR instead of the relative geographical direction of the aerodrome of departure.

6. Conclusion

In this bachelor's thesis we have analyzed the principles of implementation and performances of three different prediction models such as Probability distribution, Probabilistic Neural Networks and LSTM network.

The data used for each of these models were different in nature. The datasets used for implementation of Probability distribution model and the PNN included geographical and physical parameters, such as latitude, longitude, altitude, and speed. The LSTM model was implemented using a dataset without any geographical parameters and was based on the temporal ones. Fine-tuning experiments were done for the purpose of further examination of a non-numerical parameter – Aerodrome of departure, to test the hypothesis that this parameter can be significant for training and aid the performance of our model.

The prediction models using geographical data generally demonstrated a better performance with the error being about twice smaller than the LSTM method. The best performing model out of three analyzed was the PNN using the location and distance data.

The performance of the LSTM was not significantly improved in fine-tuning experiments possibly due to the non-numerical parameter – direction of the Aerodrome of departure - being grouped generically by geographical bearings was not specific enough. From the conclusions of the other two analyzed models, we can deduce that the direction or the origin aerodrome by itself might not hold much regressive value, but other underlying correlation caused by implicated sub-parameters must be present. One of the most significant parameters is the distance of the flight, which could be the reason why in some fine-tuned experiments there was a slight improvement in performance, as there can be a correlation between the parameters aerodrome of departure and flight distance.

From this we can conclude that the LSTM model does not perform better than alternative models in use, based on geographical radar data, however, merging functionalities of these models could bring a potential improvement in terms of performance and aid in flight planning and ground resource allocation activities.

7. Sources

[1] EUROCONTROL STANDARD DOCUMENT. Surveillance Data Exchange - Part 9, SDPS Track Messages. 1.3, European Organisation For The Safety Of Air Navigation, duben 2005.

[2] Kittler, Mojmir. (2012). Pravdepodobnostní predikce času dosažení význačných bodů v trajektorii letadel. In Bakalářská práce

[3] Hroššo, Petr. (2012) Aplikace neuronových sítí pro predikci času dosažení význačných bodů v trajektorii letadel. In Bakalářská práce

[4] Dupond, Samuel (2019). "A thorough review on the current advance of neural network structures". Annual Reviews in Control. 14: 200–230.

[5] Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.

[6] Priestley, M. B. (1982). Spectral Analysis and Time Series. London, New York: Academic Press. ISBN 978-0125649018

[7] Kingma, Diederik & Ba, Jimmy. (2014). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations.

[8] (https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.f_oneway.html

[9] Shapiro, S. S. & Wilk, M.B (1965). An analysis of variance test for normality (complete samples), Biometrika, Vol. 52, pp. 591-611.

[10] (https://www.fd.cvut.cz/personal/nagyivan/Statistika/P5_KrusWallis.pdf).

[11] https://par.nsf.gov/servlets/purl/10186768

[12] (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8459779/

[13] W. H. Kruskal & W. W. Wallis, “Use of Ranks in One-Criterion Variance Analysis”, Journal of the American Statistical Association, Vol. 47, Issue 260, pp. 583-621, 1952.

[14] G.H. McDonald, “Handbook of Biological Statistics”, One-way ANOVA.

[15] http://www.biostathandbook.com/onewayanova.html

Appendix 1

1. Python scripts made in the practical part:

`lstm.py` – *the LSTM network script. Includes fine-tuning*

`aerodromes_data_viz.py` – *script for visualization of time-series development and statistical tests on the data grouped by aerodromes*

`autocorrelate.py` – *autocorrelation test script*

`dataset.py` – *script for filtration of the raw data and creation of a dataset*

2. Raw data files in CSV format.

Each request extracts a separate CSV file. Requests were sent every 5 minutes.

`Arrivals_LKPR.zip`

Appendix 2

- **Example of a JSON request** *(requesting flight entries within the defined time window containing defined parameters)***:**

```
# Setup time window
unow = datetime.datetime.utcnow()  # UTC now
ONE_HOUR = datetime.timedelta(hours=1)  # One hour constant
ufrom = (unow - 0*ONE_HOUR).strftime('%Y-%m-%d %H:%M')  # 'From' time in
UTC
uto = (unow + 1*ONE_HOUR).strftime('%Y-%m-%d %H:%M')   # 'To' time in UTC

# Python dictionary used because of easy way to convert it to
# JSON string useing 'json.dumps' method
request = {
    "b2b": {
        "flight_data_req": {
            "airspace": "LK",
            "from": ufrom,
            "to": uto,
            "payload": {
                "item": [
                    "flightState",
                    "aircraftType",
                    "estimatedTimeOfArrival",
                    "actualTimeOfArrival",
                    "calculatedTimeOfArrival"
                ]
            }
        }
    }
}
# Convert dict to JSON string
request_string = json.dumps(request)
```

- **Example of a JSON result** *(result contains the entries of individual flights with their attributes which were defined in the request)***:**

```
JSON Result:
{
    "b2b": {
        "flight_data": {
            "head": {
                "reqTime": "2022-06-22T07:40:30Z",
                "repTime": "2022-06-22T07:40:30Z",
                "status": "OK"
            },
            "data": {
                "flights": [
                    {
                        "flight": {
                            "flightId": {
                                "id": "AT02466421",
                                "keys": {
                                        "aircraftId": "OKALT",
                                        "aerodromeOfDeparture": "LKTB",
                                        "nonICAOAerodromeOfDeparture": false,
                                        "airFiled": false,
                                        "aerodromeOfDestination": "LGSA",
                                        "nonICAOAerodromeOfDestination": false,
                                        "estimatedOffBlockTime": "2022-06-22
07:35"
                                    }
                                },
                                "aircraftType": "P46T",
                                "estimatedTimeOfArrival": "2022-06-22 10:54",
                                "calculatedTimeOfArrival": "2022-06-22 11:13",
                                "flightState": "FILED_SLOT_ISSUED"
                            }
                        }
```

- **Example of a XML request** *(requesting flight entries within the defined time window containing defined parameters)***:**

```
request = f"""<?xml version="1.0"?>
    <!-- 'FlightListByAerodromeRequest' type of request -->
    <flight:FlightListByAerodromeRequest
xmlns:flight="eurocontrol/cfmu/b2b/FlightServices">
    <endUserId>tcm</endUserId>
    <sendTime>{unow.strftime('%Y-%m-%d %H:%M:%S')}</sendTime>
    <dataset>
        <type>OPERATIONAL</type>
    </dataset>
    <includeProposalFlights>false</includeProposalFlights>
    <includeForecastFlights>false</includeForecastFlights>
    <trafficType>DEMAND</trafficType>
    <trafficWindow>
        <wef>{ufrom}</wef>
        <unt>{uto}</unt>
    </trafficWindow>
    <!-- list fileds we are interest in -->
    <requestedFlightFields>flightState</requestedFlightFields>
    <requestedFlightFields>aircraftType</requestedFlightFields>
    <requestedFlightFields>estimatedTimeOfArrival</requestedFlightFields>
    <requestedFlightFields>actualTimeOfArrival</requestedFlightFields>
    <requestedFlightFields>calculatedTimeOfArrival</requestedFlightFields>
    <countsInterval>
        <duration>0001</duration>
        <step>0001</step>
    </countsInterval>
    <!-- Arrival to LKPR airport -->
    <aerodrome>LKPR</aerodrome>
    <aerodromeRole>ARRIVAL</aerodromeRole>
</flight:FlightListByAerodromeRequest>
"""
```

- **Example of a XML result** *(result contains the entries of individual flights with their attributes which were defined in the request)***:**

```
<?xml version="1.0" ?>
<fl:FlightListByAerodromeReply
xmlns:ns18="http://www.eurocontrol.int/nm/fixm/app/ffice/1.0"
xmlns:ns17="http://www.eurocontrol.int/nm/fixm/ext/1.4"
xmlns:ns16="http://www.eurocontrol.int/nm/fixm/ext/1.3"
xmlns:ns15="http://www.fixm.aero/flight/4.2"
xmlns:ns14="http://www.fixm.aero/base/4.2"
xmlns:ns13="http://www.w3.org/1999/xlink"
xmlns:ns12="http://www.fixm.aero/messaging/4.1"
xmlns:ns11="http://www.fixm.aero/nm/1.1"
xmlns:ns10="http://www.fixm.aero/nm/1.0"
xmlns:ns9="http://www.fixm.aero/nm/1.2"
xmlns:ns8="http://www.fixm.aero/flight/4.1"
xmlns:ns7="http://www.fixm.aero/base/4.1"
xmlns:cm="eurocontrol/cfmu/b2b/CommonServices"
xmlns:as="eurocontrol/cfmu/b2b/AirspaceServices"
xmlns:fw="eurocontrol/cfmu/b2b/FlowServices"
xmlns:fl="eurocontrol/cfmu/b2b/FlightServices"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <requestReceptionTime>2022-04-20 13:09:07</requestReceptionTime>
    <requestId>B2B_CUR:12154058</requestId>
    <sendTime>2022-04-20 13:09:07</sendTime>
    <status>OK</status>
    <data>
        <flights>
            <flight>
                <flightId>
                    <id>AT00632032</id>
                    <keys>
                        <aircraftId>WZZ2752</aircraftId>
                        <aerodromeOfDeparture>LIRN</aerodromeOfDeparture>

<nonICAOAerodromeOfDeparture>false</nonICAOAerodromeOfDeparture>
                        <airFiled>false</airFiled>

<aerodromeOfDestination>LKPR</aerodromeOfDestination>

<nonICAOAerodromeOfDestination>false</nonICAOAerodromeOfDestination>
                        <estimatedOffBlockTime>2022-04-20
11:25</estimatedOffBlockTime>
                    </keys>
                </flightId>
                <aircraftType>A21N</aircraftType>
                <estimatedTimeOfArrival>2022-04-20
13:14</estimatedTimeOfArrival>
                <actualTimeOfArrival>2022-04-20 13:15</actualTimeOfArrival>
                <flightState>ATC_ACTIVATED</flightState>
            </flight>
        </flights>
```