A hybrid machine learning model for short-term estimated time of arrival prediction in terminal manoeuvring area

Zhengyi Wang, Man Liang, Daniel Delahaye

A Hybrid Machine Learning Model for Short-term Estimated Time of Arrival Prediction in Terminal Manoeuvring Area

Zhengyi Wang[a,*], Man Liang[a], Daniel Delahaye[a]

[a]*Ecole National de l'Aviation Civile,*
7, Avenue Edouard Belin, 31055 Toulouse, France

Abstract

4D trajectory prediction is the core element of future air transportation system, which is intended to improve the operational ability and the predictability of air traffic. In this paper, we introduce a novel hybrid model to address the short-term trajectory prediction problem in Terminal Manoeuvring Area (TMA) by application of machine learning methods. The proposed model consists of two parts: clustering-based preprocessing and Multi-Cells Neural Network (MCNN)-based prediction. Firstly, in the preprocessing part, after data cleaning, filtering and data re-sampling, we applied principal Component Analysis (PCA) to reduce the dimension of trajectory vector variable. Then, the trajectories are clustered into several patterns by clustering algorithm. Using nested cross validation, MCNN model is trained to find out the appropriate prediction model of Estimated Time of Arrival (ETA) for each individual cluster cell. Finally, the predicted ETA for each new flight is generated in different cluster cells classified by decision trees. To assess the performance of MCNN model, the Multiple Linear Regression (MLR) model is proposed as the comparison learning model, and K-means++ and DBSCAN are proposed as two comparison clustering models in preprocessing part. With real 4D trajectory data in Beijing TMA, experimental results demonstrate that our proposed model MCNN with DBSCAN in preprocessing is the most effective and robust hybrid machine learning model, both in trajectory clustering and short-term 4D trajectory prediction. In addition, it can make an accurate trajectory prediction in terms of Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) with regards to comparison models.

Keywords: Air Traffic Management, 4D Trajectory Prediction, Clustering, Multi-Cells Neural Network, Machine Learning, Data mining

---

☆This paper is an extension of work originally reported in SESAR Innovation Days 2017 by the authors Wang et al. (2017).

*Corresponding author
Email address: zhengyi.wang@alumni.enac.fr (Zhengyi Wang)

List of Figures

List of Tables

1. Introduction

4D trajectory prediction is the core element of future air transportation system (FAA, 2010). During the development of Trajectory Based Operation (TBO) concepts of Single European Sky ATM Research (SESAR) and Next Generation Air Transportation System (NextGen) programs, to improve the predictability of air traffic is a crucial and fundamental research topic. 4D trajectory prediction refers to the calculation and prediction of longitude, latitude, altitude and time on the future waypoint sequence based on the historical flight data. According to the time scale, 4D trajectory prediction can be divided into two categories (Guan et al., 2014):

1. Tactical (short-term) trajectory prediction: A prediction in a short period within several minutes or even shorter. Since the prediction scale is relatively small, minor change may have significant impact on prediction results. Therefore, tactical trajectory prediction requires as much information as possible. Flight-related information contained in radar or ADS-B data is usually taken;
2. Strategical (long-term) trajectory prediction: A kind of prediction before departure based on the flight plan, which provides the prediction from a macroscopic view. It is mostly applied to fuel consumption and airspace flow evaluation.

4D trajectory prediction can be influenced by numerous factors, such as aircraft weight, pilot actions, wind and temperature. These uncertainties will not only make it difficult to improve the prediction accuracy, but also will decrease the prediction process efficiency as the prediction time becomes longer (Tastambekov et al., 2014). In this paper, we propose a novel short-term trajectory prediction model, which combines different machine learning techniques, to address the problem of 4D trajectory prediction in Terminal Manoeuvring Area (TMA). The model is mainly divided into 2 parts: preprocessing model and prediction model. The preprocessing model extends and improves the trajectory clustering method proposed by Gariel et al. (2011), plays a role of preparing data with high quality for the prediction part. In addition, it supports an efficient and accurate prediction process. It contains the following steps: data cleaning, filtering, re-sampling, Principal Component Analysis (PCA), clustering and model training. In the prediction model, a novel technique named Multi-Cells Neural Networks (MCNN) is proposed. It can make parallel trajectory prediction for various shapes of traffic patterns. With Neural Network (NN) for each prediction cell, an accurate prediction for the whole complex trajectories in TMA is realised. Thus, the main contributions of this paper are threefold:

1. A novel hybrid 4D trajectory prediction model combining clustering-based preprocessing and MCNN is developed. A general guideline for selecting the model structures is presented in detail.
2. The proposed model is robust with regards to the data source. The 4D trajectory in a specified airspace containing noise and abnormal trajectories can be effectively and efficiently processed by the preprocessing model, providing high-quality inputs to the prediction model.
3. The preprocessing step leads to a substantial increase in the accuracy of the prediction model. The prediction model based on NN has a much higher accuracy than the widely-used classical linear regression model in our 4D trajectory prediction task. A comparison study is conducted to demonstrate the effectiveness of the hybrid machine learning models. Comparisons are made on K-means++ and DBSCAN algorithms in the preprocessing part, Multiple Linear Regression (MLR) and NN model in the prediction part, as well as the combination of two parts.

2. Literature review

4D trajectory prediction has attracted numerous attention from worldwide researchers in the past decades. Considerable research efforts have been taken to enrich the 4D trajectory prediction approaches. From the view of methodology, the trajectory prediction models can be mainly classified into aircraft performance models and machine learning models (Hrastovec and Solina, 2016).

Aircraft performance models belong to physics-based approaches. The model structure is based on kinematic assumptions. The model parameters are determined based on a model of the aircraft performance,

the planned flight routes, the predicted atmospheric condition, and the expected command and control strategies given by pilots or Flight Management System (FMS) (known as Aircraft Intent). The most precise aircraft performance model is Base of Aircraft Data (BADA), which provides increased levels of precision in aircraft performance parameters for modelling and simulation Nuic et al. (2010). A variety of researches based on BADA and Aircraft Intent has been conducted. For example, Xi et al. (2008) presented a classified ADS-B-based trajectory prediction algorithm. Based on the state estimation by Kalman filter and intent information captured by a pretreatment and probability method, the aircraft trajectory can be predicted with computation efficiency and fewer errors. Porretta et al. (2008) presented a novel aircraft performance model in consideration of the effect of wind, for aircraft lateral guidance and a new procedure for speed estimation. The model input includes navigation data and aircraft intent information, based on BADA set. Simulation results show that the model is suitable for reliable trajectory prediction. Kaneshige et al. (2014) described the implementation and evaluation of a motion-based trajectory prediction function, which can increase the resiliency and robustness of TBO. Based on the performance index, such as the fuel consumption, flight time, the algorithm computes the difference between with trajectory prediction and without trajectory prediction. Clearly, these aircraft performance models have made significant contributions to trajectory prediction, however, most of these models made ideal assumptions, rarely considering the actual constraints and human behaviour factors. The prediction in this method is conducted one by one, which cannot reflect the interaction between trajectories. In addition, once the data resources are limited or poorly supported, the prediction models will be ineffective for short-term trajectory prediction, not to mention large-scale trajectory prediction problems.

As a branch of Artificial Intelligence (AI), machine learning has been developed over 30 years. One of the major function of AI is learning from experience and make predictions. The trend of recent years shows that machine learning is commonly used in trajectory prediction domain. Compared with those aircraft performance models, machine learning models were built with weak assumptions or even without assumptions. In most cases, it shows better prediction performance. For example, Le Fablec and Alliot (1999) used Neural Networks (NN) to predict an aircraft trajectory in the vertical profile. Trained by real historical trajectory dataset, 2 different model structures were presented. In the first method, the inputs include current altitude, the remaining altitude to reach Request Flight Level (RFL) and n past vertical speeds. The output is the next speed. The inputs of the second structure consist of starting altitude and remaining altitude to reach the RFL. n first initial speeds is the output. Simulation result showed that NN model gives better results than classical prediction functions based on the model of aircraft. Leege et al. (2013) introduced Generalized Linear Model (GLM) for trajectory prediction at prediction horizons of 15NM to 45NM on fixed arrival route. The inputs of this model are aircraft type, ground speed at Initial Approach Fix (IAF), altitude over IAF, surface wind and altitude wind. All inputs come from surveillance data and meteorological data. Alligier et al. (2015) also applied GLM to predict the airspeed during climb. The simulation is conducted on actual Mode-C radar data of 9 different aircraft types. The prediction error is quite small, compared with BADA reference speed profiles.

In view of improving the accuracy in prediction tasks, Trivedi et al. (2015) carried out a study on the feasibility of utilizing clustering as a preprocessing approach. Their research demonstrates that the improvement on prediction accuracy is significant on large-scale clusterable datasets by combining the clustering with even some simple machine learning predictors. Under routine traffic situation in the TMA, the aircraft follows the standard arrival/departure procedure and regular ATC instructions, which makes trajectories clusterable. Thus, the application of machine learning together with clustering for 4D trajectory prediction is a valuable and interesting research topic. Numerous clustering models have been proposed for airspace monitoring, traffic patterns identification, automated traffic analysis, trajectory efficiency analysis (Eckstein, 2009; Gariel et al., 2011; Chen et al., 2011; Tang et al., 2015; Ayhan and Samet, 2015; Nicol and Puechmorel, 2016).

Based on these studies, models combining clustering with simple predictors have been investigated in the latest years. Tastambekov et al. (2014) considered the short to mid-term aircraft prediction problem, namely, the prediction with a time horizon from 10 min to 30 min. The model firstly searches similar trajectories in terms of shape and time, then uses wavelet decomposition to solve the linear regression model of the relationship between time and trajectory projection onto one of the three axis X, Y and Z. This

method gets good prediction results with high robustness and efficiency. Hong and Lee (2015) introduced a new framework for predicting Estimated Time of Arrival (ETA) of aircraft from a specified entry fix. The model contains two steps: trajectory pattern identification by Dynamic Time Warping (DTW) and MLR models training for each pattern. The prediction of arrival time can be achieved by applying different MLR model for each trajectory pattern of target aircraft. However, most of the existed clustering-based prediction model are unable to conduct a comprehensive preprocessing step. Some models neglect the prediction steps, directly considering clustering results as prediction results. A majority of trajectory pattern identification approaches are not robust, requiring carefully selected flight data that follow the same departure/arrival procedure or pass the specified entry fix rather than in a specified airspace. If there exists some noise and overflights in the data, the predicted results will be far less effective. In addition, in the existed prediction model, the regression models that have been used are relatively simple and have a shallow structure. What's more, for different traffic patterns, the prediction processes have to be executed successively, without parallel computing ability.

3. Methodology

In order to improve the prediction accuracy and efficiency, we propose a novel hybrid trajectory prediction approach, see Fig. 1. There are two parts in this model: clustering-based preprocessing model and MCNN-based prediction model. The preprocessing step is mainly composed of re-sampling, data augmentation, PCA and DBSCAN algorithm. In this part, the model firstly cleans and reshapes the data, then identifies the 4D trajectories into different clusters and removes noise in an efficient way. Each cluster symbolizes that the corresponding trajectories have the similar pattern. Noise contains trajectories with holding patterns, trajectories with large vectoring, trajectories in special cases and overflight trajectories. Finally, the trajectory data quality will be highly increased. In the prediction step, we apply the MCNN model for 4D trajectory prediction. For each partition of trajectories, there is a predictor, in which there is an individual NN-based learning cell. Each individual learning cell will be trained with the associated cluster of trajectories. Consequently, each cluster of trajectories will have its corresponding prediction model. For the new input trajectory, we will classify them into different corresponding clusters by decision trees according to the entry point, then with already trained multi-cells predicting model, trajectory prediction of the input data will be generated.

3.1. Data preparation

We choose Automatic Dependent Surveillance-Broadcast (ADS-B) as the data source of the 4D trajectory prediction model. Each record of ADS-B data contains:

1. Type of operation (departure/arrival),
2. Coordinated Universal Time (UTC) time stamp t,
3. Aircraft number,
4. Position (X, Y, Z),
5. Heading Ψ,
6. Horizontal ground velocity V_h,
7. Vertical ground velocity V_v, etc.

Each record with the same aircraft number belongs to an aircraft i, and the collection of all records for that aircraft forms the trajectory T_i, $i = 1, ..., n$, where n is the total number of trajectories in the dataset.

3.2. Clustering-based preprocessing

3.2.1. Data Cleaning and Formatting

To start with, the dataset is filtered on recording time scale, recording length and so on, making sure the data validity. Due to the instability of ADS-B receiver, the collected ADS-B data may not be continuous.

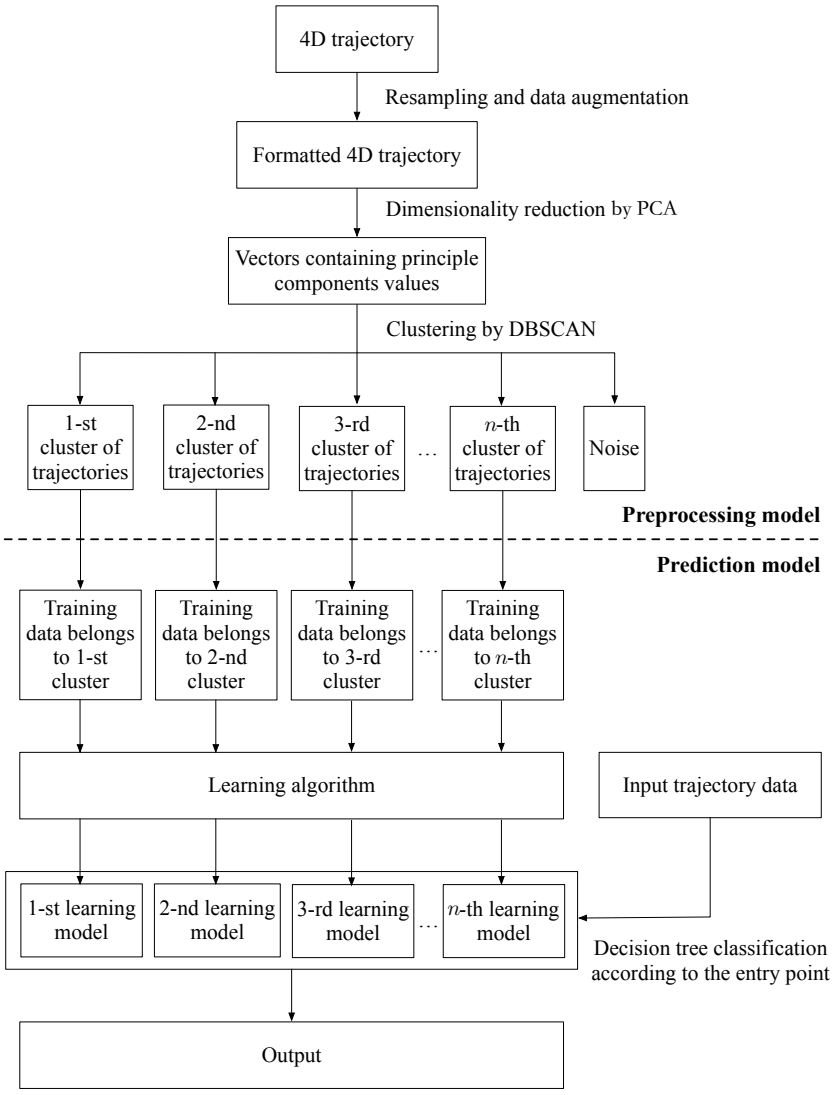


Figure 1: Proposed 4D trajectory prediction approach

Some trajectories may have missing parts. To solve this problem, it is necessary to place a low pass filter with the following function:

$$\tilde{x}_{1,i} = x_{1,i} \tag{1}$$

$$\tilde{x}_{l,i} = \alpha x_{l,i} + (1-\alpha)\,\tilde{x}_{l-1,i}, \quad l = 2, ..., m_i - 1. \tag{2}$$

where the aircraft position in 3D coordinates and the heading of the l-th point of i-th trajectory are substituted into $x_{l,i}$. α is a smoothing factor in $[0, 1]$. In this study, α is set to 0.5 to provide better results without too much delay. m_i is the number of points in i-th trajectory.

In order to make dataset suitable for clustering, each trajectory should be represented as a vector. All the trajectory vectors are resampled into the same length, then their distance (which will be discussed in the following section) can be computed. The resampling method for i-th trajectory is given as follows:

$$T_i = \left\{ T_{l,i} \,\middle|\, l = \text{floor}\left(\frac{k \cdot m_i}{S}\right), k = 1, ..., S \right\} \tag{3}$$

where the function floor(x) takes the largest integer that does not exceed x. l is a set containing all the index of sampled points for each trajectory, S is the number of sampled points for each trajectory. Experience shows that $S = 50$ would have been enough for clustering purpose. Trajectories with less than 50 points were eliminated due to statistical insufficiency.

3.2.2. Dimensionality augmentation

This step aims to augment the dimensionality of dataset for better clustering. The existing dimensions may not be sufficient and will result in lack of information, which can't completely reflect the differences between each trajectory. Thus, augmentation of dimensions will help improve the clustering performance. Therefore, the following dimensions will be added into the dataset:

1. Distance from the reference point R, which indicates the convergence of aircraft toward the center of the TMA. Due to the runway configuration, we can define the reference point $(X_{\text{ref}}, Y_{\text{ref}}, Z_{\text{ref}})$ as the centre of the runway. For each trajectory point, $R_{l,i}$ is given as:

$$R_{l,i} = \sqrt{(X_{l,i} - X_{\text{ref}})^2 + (Y_{l,i} - Y_{\text{ref}})^2 + (Z_{l,i} - Z_{\text{ref}})^2} \tag{4}$$

2. Distance from the corner point D. The corner point $(X_{\text{cor}}, Y_{\text{cor}}, Z_{\text{cor}})$ can be assigned as a corner of the considered airspace. The corner point will help solve the identifying problem when two trajectories are symmetric. The $D_{l,i}$ is calculated by the function below:

$$D_{l,i} = \sqrt{(X_{l,i} - X_{\text{cor}})^2 + (Y_{l,i} - Y_{\text{cor}})^2 + (Z_{l,i} - Z_{\text{cor}})^2} \tag{5}$$

   The reference point and corner point play the role as multilateration.
3. Angular position from the reference point Θ. It shows the variation (turning status) of trajectories with respect to the reference point. Θ is defined as:

$$\Theta_{l,i} = \arctan\left(\frac{Y_{l,i} - Y_{\text{ref}}}{X_{l,i} - X_{\text{ref}}}\right) \tag{6}$$

To sum up, the resampled dataset includes original features: UTC time stamp t, position (X, Y, Z), heading Ψ and additional features: distance from the reference point R, distance from the corner point D, angular position from the reference point Θ. To avoid the discontinuity at $\pm\pi$ and for feature scaling, the sine and cosine values of Θ and Ψ is utilised instead of the original value.

Next, to make every feature on the same scale, each feature is normalised in $[0, 1]$. The general formula is given as:

$$x^* = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{7}$$

where x is the original feature and x^* is the normalised feature. Replacing x with our features, finally, all features of each trajectory is concatenated and organised as follows:

$$T_i = [t_i^* \;\; P_i^* \;\; R_i^* \;\; D_i^* \;\; \cos(\Theta)^* \;\; \sin(\Theta)^* \;\; \cos(\Psi)^* \;\; \sin(\Psi)^*] \tag{8}$$

$$T = \begin{bmatrix} T_1 \\ \vdots \\ T_n \end{bmatrix} \tag{9}$$

where $P_i^* = [X_i^* \;\; Y_i^* \;\; Z_i^*]$. Then, each trajectory is reshaped into a vector with 500 components. Matrix T is $n \times 500$ and contains all trajectories with features, where n is the total number of trajectories.

3.2.3. Principal Component Analysis

As shown in Eq. (8), trajectories are related to various of factors. Nevertheless, among these factors, some are more related, while the other is less related. Redundant elements will decrease computational

efficiency, even lead to larger errors. To solve this problem, PCA is introduced. PCA is a powerful tool used to reduce the dimension of dataset without losing too much information. The main idea of PCA is to derive an orthogonal linear transformation to project each of the vector variables into principal components for the maximum amount of variance that can be presented in lower dimensions (Runkler, 2012).

PCA performs a linear transformation on the $n \times m$ (in this case $m = 500$) matrix T:

$$Y = E \cdot T \tag{10}$$

where E is a rotation matrix, Y is the new principal component matrix. The variance of Y is:

$$\mathrm{var}(Y) = E^T \cdot C \tag{11}$$

where C is the covariance matrix of T, which can be written as:

$$C = \frac{1}{n-1} \cdot T \cdot T^T \tag{12}$$

The eigenvalues of C can be calculated as $\{\lambda_i | i = 1, ..., m\}$, which correspond to the variances in Y as $\{v_i | i = 1, ..., m\}$, with $\lambda_1 > \lambda_2 > ... > \lambda_n$.

To map a dataset $X \subset \mathbb{R}^m$ to a dataset $Y \subset \mathbb{R}^q$ with $q \in \{1, ..., m-1\}$, a rotation matrix $E = (v_1, ..., v_q)$ can be used. The dimension can be reduced by choosing the number of q. In order to keep most of the information while reducing dimensions effectively, it is required that the projection should better covers 90% of the variances, i.e., the cumulative percentage or variance explained $G(q)$ is greater than 90%:

$$G(q) = \frac{\sum_{i=1}^{q} \lambda_i}{\sum_{i=1}^{m} \lambda_i} \geq 90\% \tag{13}$$

3.2.4. Clustering

Clustering algorithm belongs to unsupervised learning, for the purpose of partitioning unlabelled data according to similarity. The widely used clustering algorithm for air traffic pattern identification is K-means (Eckstein, 2009; Tang et al., 2015) and DBSCAN (Gariel et al., 2011; Chen et al., 2011), along with their improved or combination version.

The K-means algorithm (MacQueen et al., 1967) is an iterative clustering algorithm that partitions N observations into K clusters, where the number of clusters K is a hyperparameter. The K-means algorithm aims to minimise the Within-Cluster Sum of Squares (WCSS) cost function.

$$\mathcal{L}(S; \mu) = D_K = \sum_{i=1}^{K} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2 \tag{14}$$

where S is the set of clusters partitioned by this algorithm, μ_i is the mean of points in S_i and $K \leq N$. Initialisation of centroid seeds demands a-priori knowledge. The common idea is to use a farthest-first traversal approach or a selection at random from the dataset. Nevertheless, this is not robust to outliers. Instead, Arthur and Vassilvitskii (2007) proposed K-means++ algorithm using a heuristic to initialise the centres. It improves both the computation speed and accuracy of K-means.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a commonly used density-based clustering algorithm (Ester et al., 1996). The core concept of DBSCAN is to evaluate the density depending on the number of points within the ϵ-neighbourhood. DBSCAN classifies the observation points into three types: core point, density-reachable point and noise point. The algorithm expands to density-reachable areas from a selected core point, then obtaining a maximum area including the core point and density-reachable points. Being robust to the quality of datasets, DBSCAN can divide the dataset into several clusters and noise, where the a-priori selection of the number of clusters is not required. Besides,

DBSCAN is able to find arbitrarily shaped clusters. The advantages of DBSCAN make it fit well with trajectory clustering scenarios. There are two hyperparameters in DBSCAN algorithm: the neighbourhood radius ϵ and the minimum number of points required to form a dense region $minPts$. The value of ϵ affects the size of clusters. The value of $minPts$ affects the noise identification and the significance of clusters. After clustering, each cluster of trajectories can be considered as a main traffic pattern managed by controllers, and it has been passed by a large number of flights. The noise represents trajectories in some special cases, such as holding patterns and large vectoring.

In our case, the first q principal components of the organised trajectory matrix, computed by equation (13), are clustered by machine learning algorithm, DBSCAN or K-means++.

3.3. MCNN-based prediction model

In this paper, supervised learning techniques are used for short-term trajectory prediction. Each input x is a vector that contains the 3D position, heading, horizontal speed and vertical speed. The target variable t is the Estimated Time of Arrival (ETA) on the runway. We consider the following fixed-design regression model:

$$Y_n = f(x_n) + \epsilon_n, \quad n = 1, ..., N \tag{15}$$

where Y_n are the random variable that follows a mean function $f(\cdot)$ with errors ϵ_n, which are independent and identically distributed (i.i.d.) random variables, such that $E(\epsilon_n) = 0$ and $\text{var}(\epsilon_n) = \sigma^2$. The most classical machine learning model Multiple Linear Regression (MLR) aims to estimate the function $f(\cdot)$ under a linear assumption. Given multiple independent variables of N statistical units with D dimensions $x_n = (x_{n1}, ..., x_{nD})^T$ and corresponding dependent variable y_n, $n = 1, ..., N$, the goal is to estimate the mean function $f(\cdot)$ by $h_w(\cdot)$:

$$h(x; w) = \sum_{d=1}^{D} w_d x_{nd} + w_0, \quad n = 1, ..., N \tag{16}$$

where $\{w_n | n = 0, ..., N\}$ are coefficients, which can be approximated by least squares regression, taking the estimator $h_w(\cdot)$ that minimises the Mean Squared Error (MSE).

In our case, we use MCNN model to predict the ETA based on preprocessed real 4D trajectory data. For each cluster (cell) generated by the training set, an individual NN-based learning model will be built by training the dataset in the associated cluster. Consequently, each cluster will have an individual learning model. Then, the test set will be classified into different cell by classification algorithm. The output will be produced by the corresponding NN-based learning model. The advantage of using Neural Network (NN) in each cell is that they are able to learn the hidden and non-linear dependencies from the training data. The architecture of the proposed NN model for each cell is composed of an input layer, a hidden layer and an output layer, shown in Fig. 2. Given C cells, for each cell c, $c \in \{1, ..., C\}$ and the input vector $(x_1, ..., x_{N_c})^T$ in test set T_c, the output can be calculated as:

$$h_c(x; w) = \Psi\left(\sum_{i=1}^{M_c} w_{i,c} \Phi\left(\sum_{j=1}^{N_c} w_{ij,c} x_j + w_{0j,c}\right) + w_{0,c}\right) \tag{17}$$

where M_c is the hidden layer node number, $w_{ij,c}$ is the weight between the j-th input node and the i-th hidden node, $w_{i,c}$ is the weight between the i-th hidden node and the output node, $w_{0j,c}$ is the bias to the i-th hidden layer, $w_{0,c}$ is the bias to the output layer, Φ is the sigmoid logistic function:

$$\Phi(z) = \frac{1}{1 + e^{-z}} \tag{18}$$

Ψ is the identity function:

$$\Psi(z) = z \tag{19}$$

To reduce over-fitting, the error function we are trying to minimise is:

$$E(w) = \sum_{(x,y)\in T_c} \left(h_c(x;w) - y\right)^2 + \lambda_c \sum_{j=1}^{N_c} \sum_{i=1}^{M_c} w_{ij,c}^2 \tag{20}$$

where λ_c is the regularisation coefficient for NN model in cell c. The back-propagation method was used to compute the gradient of the error function, and Levenberg-Marquardt algorithm (Reynaldi et al., 2012) was used to minimise the error.

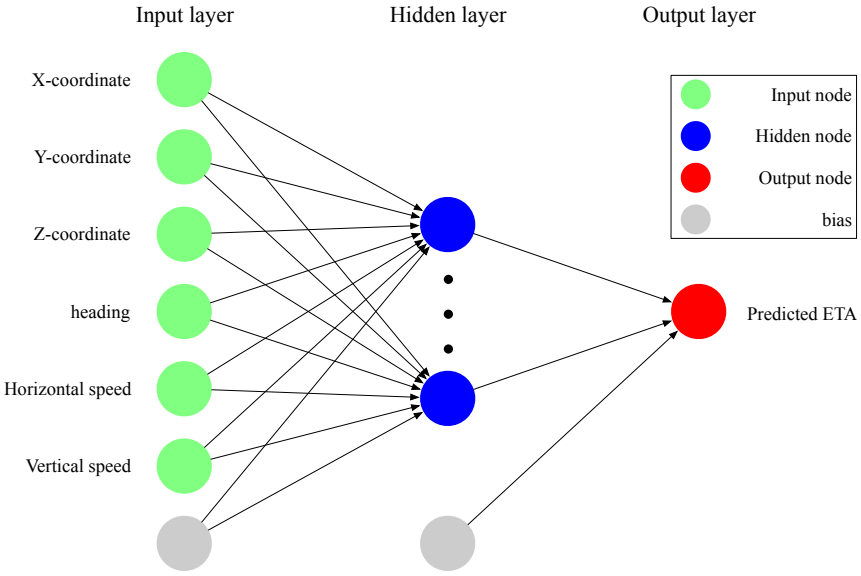


Figure 2: Neural network architecture used in this paper

The new input is classified by decision trees according to the entry point of the corresponding trajectory. The advantages of the classification decision tree are that it does not acquire any a-priori knowledge and expensive computation costs. Once the classification tree model is built, the worst time complexity of classifying an unknown sample is $\Theta(H)$, where H is the maximum depth of the classification tree. In addition, in view of arrival flights in TMA, the longitude and latitude of entry points of trajectories in each cluster belong to a certain range in the airspace. This characteristic can be used to realise an effective classification on new input. In this work, Classification And Regression Trees (CART) algorithm is utilised for the decision tree, with the aircraft position X and Y as input.

3.4. Performance evaluation and model selection

To assess the performance of MCNN model, the MLR is proposed as the comparison learning model. DBSCAN and K-means++ are compared in preprocessing clustering part, so as to test different hybrid machine learning models.

In machine learning, hyperparameter stands for the parameters that are fixed before actual learning process begins, which cannot be directly adjusted by empirical risk. The bias-variance tradeoff is controlled by hyperparameters.

Firstly, in the preprocessing part, we will evaluate the performance of clustering algorithm DBSCAN, compared with K-means++. Different hyperparameters selection approaches are proposed for these clustering algorithms. The training data used is composed of the first q principal components of trajectory matrices T_i in equation (8) that belongs to the training set of each outer cross validation fold. The number of clusters of K-means++ can be estimated by the silhouette method (Rousseeuw, 1987). The silhouette value for

each instance i is a measure of similarity between its cluster and of difference between other clusters. The silhouette value is defined as:

$$s_i = \frac{a_i - b_i}{\max(a_i, b_i)} \tag{21}$$

where a_i is the average distance from the i-th instance to the other instances of the same cluster, and b_i is the minimum average distance from the i-th instance to instances in a different cluster. For each K in the hyperparameter grid, the value that maximises the average silhouette coefficient of all instances in the training set is chosen as the optimal cluster number of K-means++. On the contrary, the number of clusters of DBSCAN algorithm need not to be determined a-priori. However, DBSCAN algorithm has 2 hyperparameters ϵ and $minPts$, in which ϵ can be evaluated by computing the K-Nearest Neighbour (K-NN) distance in the training set (Sander et al., 1998). The principal concept of this method is to calculate the distance of every point to its K nearest neighbour, and these distances are plotted in an ascending order. The optimal ϵ value corresponds to the "knee" (threshold K-distance value), which is relatively simple to be recognised in a graphical representation. Since K-NN is a non-parametric method, the value of K is specified manually. With regards to $minPts$, there is no general approach to estimate the value. In order to remove most abnormal trajectories and balance the size of dataset in the meantime, we search an optimal value of $minPts$ in the hyperparameter grid to make sure that the noise account for $5\% \sim 8\%$ of the dataset.

Secondly, in order to well select the hyperparameters of prediction model, and to achieve an unbiased performance of the prediction model, this paper uses nested cross validation method. It consists of the outer loop and the inner loop. A K_1-fold cross validation splits the dataset S into K_1 subsets S_i, $i \in \{1, ..., K_1\}$. For each outer iteration i, $K_1 - 1$ folds $S_{-i} = S \backslash S_i$ act as training sets and one fold S_i is test set. Then, there is another K_2-fold cross validation, which will further split the training sets S_{-i} into K_2 subsets $S_{-i,j}$, $j \in \{1, ..., K_2\}$. For each inner iteration j, $K_2 - 1$ folds $S_{-i} \backslash S_{-i,j}$ play the part of training sets and the remaining fold $S_{-i,j}$ is validation set. The purpose of the inner loop is the selection of hyperparameters and the outer loop aims to assess the model performance. Taking $K_1 = 5$, $K_2 = 5$, the proportion of training sets, validation sets and test sets is 64%/16%/20%. The kernel idea of the whole process can be summarised on Fig. 3.

Here, we use a grid search hyperparameters tuning algorithm (Alligier et al., 2015) on NN-based learning model. This algorithm aims at selecting the hyperparameter λ of an algorithm A_λ by performing a 5-fold cross-validation on a set of examples T, which is depicted by algorithm 1.

Algorithm 1 Hyperparameters tuning

1: **function** TuneGrid(A_λ, $grid$)[T]
2: $\quad \lambda* \leftarrow \underset{\lambda \in grid}{\arg\min}\, CV_5(A_\lambda, T)$
3: $\quad$ **return** $A_{\lambda*}[T]$
4: **end function**

The coefficient of determination R^2 and the statistical significance p-value are considered as the statistics for the error between the actual and estimated results of the MLR model. R^2 is defined as equation (22), providing a measure of how close the dependent variables are to the independent variables. The p-value tests the null hypothesis that the coefficient is equal to zero.

$$R^2 = 1 - \frac{\sum_{i=1}^{N} (y_n - \hat{y}_n)^2}{\sum_{i=1}^{N} (y_n - \bar{y})^2} \tag{22}$$

The performance of classification model is estimated by K-fold cross validation on the training set, setting $K = 5$ in this case.

Finally, we use Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to assess the

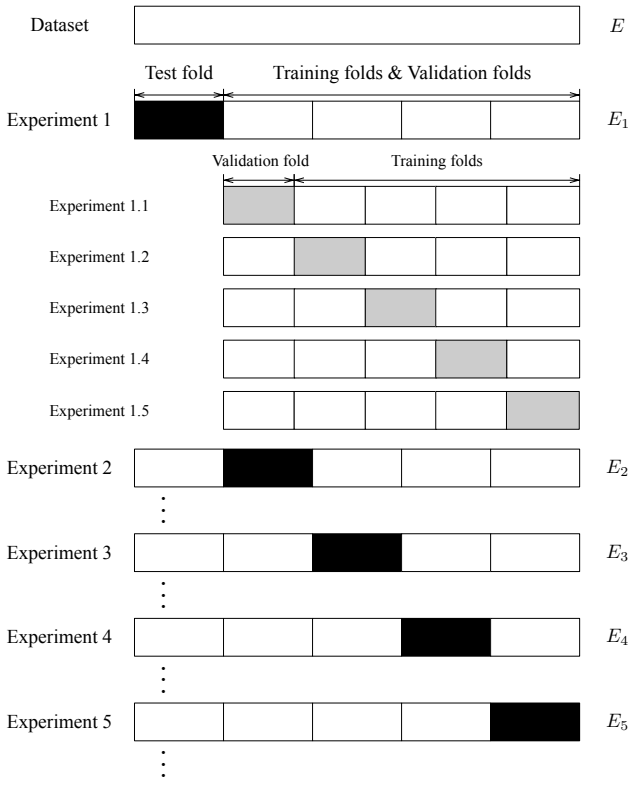


Figure 3: Nested cross validation procedure

performance of trajectory prediction models:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{23}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}, \tag{24}$$

where $\hat{y}_i$ is the i-th predicted value and y_i is the i-th observed value of ETA. Given that each outer iteration produces a $\mathrm{MAE_i}$, and a RMSE_i, $i = 1, ..., k_1$, the average MAE and RMSE can be computed as follows:

$$\mathrm{MAE} = \frac{1}{k_1}\sum_{i=1}^{k_1}\mathrm{MAE}_i, \tag{25}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{k_1}\sum_{i=1}^{k_1}(\mathrm{RMSE}_i)^2}, \tag{26}$$

4. Experiment results and discussion

4.1. Dataset

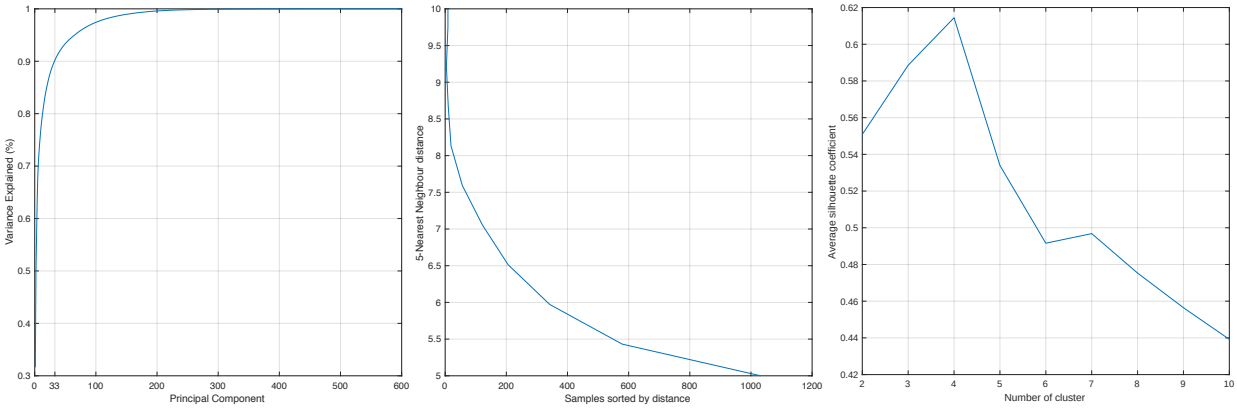
The available dataset includes ADS-B records in July, 2017 over the TMA of Beijing Capital International Airport (BCIA), which is one of the busiest airport in the world, with three parallel runways: 18R/36L,

18L/36R and 01/19. Since the studied airspace is relatively small, the longitude, latitude of trajectory points in WGS84 can be projected into a projection plane. Note that, in this paper, arrival flights that correspond to QFU 36 are taken into consideration, where QFU means the magnetic orientation of runway-in-use. QFU 36 goes to North.

The dataset that we used in the experiments contains 8,677 arrival flights of QFU, being made up of 902,457 trajectory points. After data cleaning procedure, 103 flights were removed from the dataset, and the data validity rate is higher than 98%. Then, the dataset is formatted and augmented with dimensionality, depicted in section 3.2.

4.2. Clustering performance

First, in this study, the cumulative percentage of variance of trajectory matrix in equation 8 is calculated and presented in Fig. 4a. When the principal component reaches over 33, the variance explained will exceed 90%. Let $q = 33$, then the dimension of each trajectory was reduced to 33 from 500. To sum up, dimensionality augmentation enriches the features that principal components can choose. PCA reduces the dimension of the dataset, which makes the following clustering step more efficient and accurate in the projected principal component space.



(a) The cumulative percentage of variance in PCA (b) DBSCAN hyperparameters tuning (c) K-means hyperparameters tuning

Figure 4: Parameters selection of algorithms in the preprocessing step

Table 1: The grids of hyperparameters for machine learning algorithms

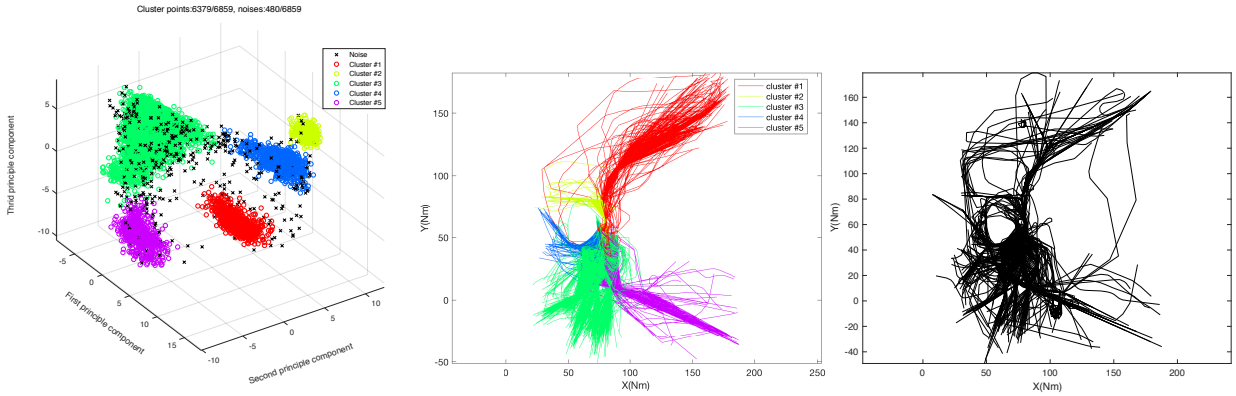
Method	Hyperparameter grid
K-means++	$K = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$
DBSCAN	$minPts = \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$
MCNN	$\lambda_c = \{0.0001, 0.001, 0.01, 0.1, 0.5, 1, 2, 5\}, \quad c = 1, ..., C$
	$M_c = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}, \quad c = 1, ..., C$

Then, the hyperparameters grids of aforementioned machine learning approaches are used, see Table 1. According to the density distribution of the dataset, K is set to 5 in K-NN. In all outer cross validation folds, K-NN distance plots of instances in the training set indicate that $\epsilon = 6.5$. Fig. 4b represents the K-NN distance plot in one of the folds. $minPts = 50$ satisfies that the noise accounts for $5\% \sim 8\%$ of the dataset. Among the hyperparameter grid of K-means++, $K = 4$ minimises the average silhouette value of all instances in the training set for every outer cross validation. Fig. 4c is extracted from one of the cross validation fold as an example.

We find that DBSCAN algorithm clusters the trajectories into 5 partitions and noise. Taken one of cross validation folds into consideration, Fig. 5a represents 3D plot for the first principal components of the trajectory matrix. In this case, cluster points involve 93.00% and the noise points account for 7.00%. Black

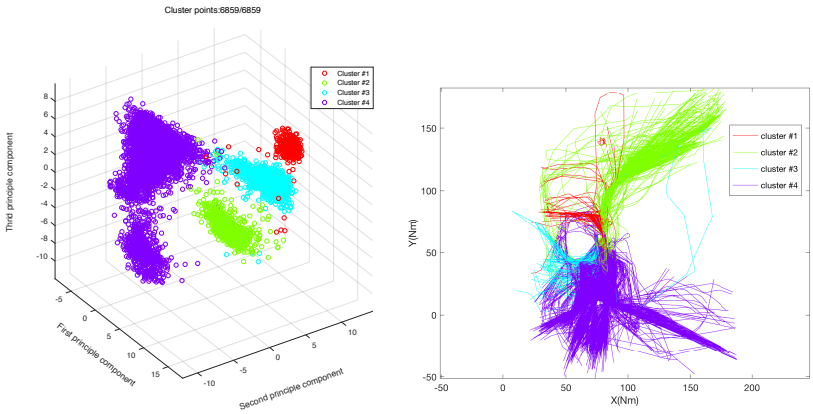
crosses indicate noise, and circles with different colour stand for trajectories in different clusters. Fig. 5b demonstrates the corresponding 2D trajectories. It can be seen that 5 main arrival procedures are clustered. Fig. 5c shows trajectories recognised as noise, which is mainly composed of holding patterns and trajectories with large vectoring, which will have an interference for the training stage. Therefore, the noise should be deleted from the dataset. With the same cross validation fold as Fig. 5, K-means++ clusters all the trajectories into 4 partitions. Unlike DBSCAN, K-means++ would not identify the noise. Fig. 6a represents the first 3 principal components of the trajectory matrix. Fig. 6b shows the corresponding trajectories, in which the cluster #4 combine cluster #3 and cluster #5 in Fig. 5b together. The clustering result of K-means++ is not good, considering the combining cluster contains several arrival procedures.

In addition, Fig. 7 illustrates the decision trees for classification according to clusters generated by DBSCAN and K-means++ in one of the outer cross validation folds. It is clear that according to maximum depth and the number of nodes, the time complexity and the overfitting risk of Fig. 7a are much smaller than Fig. 7b.



(a) DBSCAN clusters (b) Clustered trajectories (c) Trajectories identified as noise

Figure 5: DBSCAN clustering example in one of the outer cross validation folds



(a) K-means++ clusters (b) Clustered trajectories

Figure 6: K-means++ clustering example in one of the outer cross validation folds

4.3. Prediction performance

Different time horizons of prediction will have different prediction error. To depict the error distribution, box-plot is utilised as a graphical tool to visualise the main statistical measurements. Fig. 8 portrays the

(a) Classification tree for DBSCAN

(b) Classification tree for K-means++

Figure 7: Classification tree trained by DBSCAN and K-means++ clusters in one of the outer cross validation folds

model prediction performance at different time horizons. In each sub-figure, the X-axis presents the transit time $t(s)$ in TMA. While $t(s) = 0$, it means that aircraft is entering the TMA, and $t(s) = 800$ means that aircraft is flying in TMA for 800 s. Note that the shortest transit time in TMA is around 800 s, which mainly corresponds to the flights entering from west. When $t(s) > 800$, some of the flights may have landed on the runway. Fig. 8 shows the prediction errors for trajectories from all the entry points of TMA. The Y-axis depicts the absolute prediction error between predicted ETA and observed Actual Time of Arrival (ATA). For each 50 s time step, a box-plot shows the 5%, 25%, 50%, 75% and 95% quantiles. The remaining 10% error mostly come from abnormal trajectories (highly deviated, go around, with holding procedure, in emergency, etc.), which are difficult to be predicted. So this part of error is not displayed. Two models are presented in each sub-figure. For clearly visualisation, two models are overlaid. We find that with different prediction methods, the results show the same trend, that is: when the time to destination is fewer, the absolute prediction error is smaller. We further observe the distribution of ETA prediction errors. Note that the whiskers contains 90% of the prediction errors, and the box contains 50% of the prediction errors. At the entry time of TMA, $t(s) = 0$, in view of all prediction methods except MLR model, over 50% of trajectory predictions errors are less than 125 seconds. According to *Report on National Civil Flights Operation Efficiency in 2017* published by CAAC Operation Monitoring Center (2018), the average arrival delay time of BCIA is 30.7 minutes per aircraft. By comparison, the result of the proposed model is satisfactory, which can better predict delays.

Overall, the NN model prevails against the MLR model. Moreover, in view of the same machine learning method, the model with clustering preprocessing step has less prediction errors than the one without clustering preprocessing step. The preprocessing part can improve the prediction performance of the MLR model, and DBSCAN-based preprocessing performs slightly better than K-means++-based preprocessing. The preprocessing part plays a positive effect both on NN-based model and MLR-based model, in which MLR-based model has a greater improvement on prediction performance. Among all models, the NN model combined with DBSCAN preprocessing model performs best ETA prediction. Thus, we conclude that the clustering preprocessing is effective in improving the prediction accuracy on almost all time horizons, especially DBSCAN algorithm.

For different traffic patterns (clusters), the prediction error from different entry point of TMA ($t(s) = 0$) is different. The statistics of prediction results on NN and MLR with DBSCAN-based preprocessing model in each clusters and each outer cross validation fold were summarised in Tab. 2 and Tab.4. For MLR model, the correlation coefficient R^2 between the ATA and estimated ETA for each clusters is computed, representing the percentage of the variance explained in the ATA. The computed p-value for the null hypothesis was almost zero, which was small enough to reject the null hypothesis at a significance level of 0.01.

In Tab. 2, new test data can be well clustered into 5 partitions by application of the classification tree and the DBSCAN algorithm. The accuracy of the classification tree is more than 99.5%, and in view of MAE and RMSE, the proposed NN model performs better than the MLR model for each cluster. For cluster #4, the associated MAE is generally smaller than the other partitions, on account of the shape and the flying distance of the arrival procedure of trajectories. We also find that cluster #4 usually has

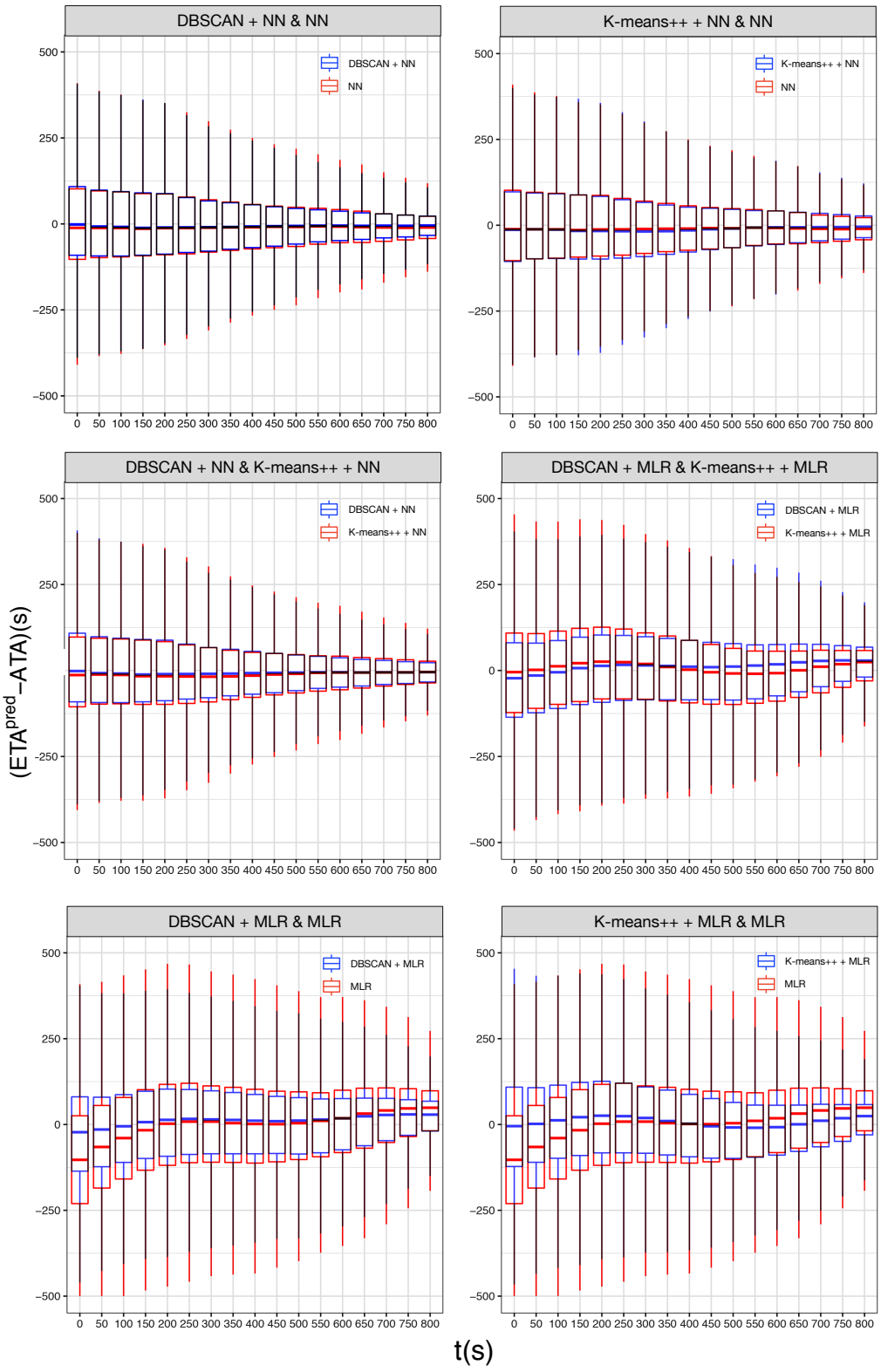


Figure 8: Box-plots that portray the error distribution on 6 models

the shortest transit time in TMA, which is from the west with less deviation before landing. Besides, in Tab. 3, compared with DBSCAN-based models, we find that the average number of nodes of classification tree for K-means++ is 11 times larger, the classification accuracy for K-means++ is about 3% lower, and the percentage variance of clusters in different outer cross validation folds of K-means++ is much less stable. Taking Fig. 5 as an example to explain the reason of such differences. In Fig. 5a and Fig. 5b, there are some clusters that intersect. In addition, because K-means++ cannot identify the noise, some abnormal or deviated trajectories were partitioned in clusters. Therefore, the classification trees have to devote more efforts for noise and clusters with intersection, which need more binary classifiers for fine classification that may cause overfitting. On the contrary, the shape of clusters generated by DBSCAN is quite good, and there is no obvious intersection between clusters. Even the decision trees for K-means++ are more complicated, the overall performance is getting worse. Compared with the performance of ETA prediction on NN and MLR with K-means++-based preprocessing step on each outer cross validation fold, It is also indicated that NN model performs still better than MLR model for each cluster in terms of MAE/RMSE. Last but not least, in Tab. 4, regarding overall prediction errors MAE and RMSE, the NN model with DBSCAN preprocessing performs best, followed by NN model with K-mean++. MLR model without preprocessing performs worst on the prediction task.

Table 2: The performance of ETA prediction on NN and MLR with DBSCAN-based preprocessing step on each outer cross validation fold

Outer CV#	Classification tree		Cluster#	Percentage	DBSCAN P. + NN		DBSCAN P. + MLR			
	Accuracy	Size			MAE[s]	RMSE[s]	MAE[s]	RMSE[s]	R^2	p-value
1	99.92%	19	1	5.15%	134.20	166.56	162.80	201.44	0.90	≈ 0
			2	12.19%	192.25	275.28	229.83	314.99	0.90	≈ 0
			3	54.98%	128.60	184.52	154.94	215.48	0.88	≈ 0
			4	11.71%	84.31	142.19	90.32	144.31	0.94	≈ 0
			5	9.26%	161.13	223.92	200.39	255.23	0.89	≈ 0
2	99.61%	15	1	5.42%	164.42	218.15	175.22	236.19	0.90	≈ 0
			2	12.03%	184.90	286.96	220.73	312.54	0.91	≈ 0
			3	55.74%	142.85	233.94	165.28	251.08	0.88	≈ 0
			4	11.21%	106.91	160.81	120.16	173.04	0.94	≈ 0
			5	9.48%	188.87	356.39	218.93	368.60	0.92	≈ 0
3	99.84%	19	1	4.93%	128.23	161.03	134.21	176.84	0.90	≈ 0
			2	10.98%	161.44	203.57	236.30	287.21	0.91	≈ 0
			3	56.66%	144.49	211.19	162.70	228.78	0.88	≈ 0
			4	10.76%	81.07	108.45	113.08	144.20	0.94	≈ 0
			5	9.40%	146.32	356.39	195.29	249.01	0.89	≈ 0
4	99.77%	21	1	4.99%	122.03	155.63	153.30	202.44	0.90	≈ 0
			2	11.12%	160.28	218.39	198.74	268.66	0.91	≈ 0
			3	57.59%	123.23	173.47	137.96	183.15	0.88	≈ 0
			4	10.50%	95.19	151.39	154.52	209.90	0.94	≈ 0
			5	9.86%	139.49	293.05	185.23	330.03	0.88	≈ 0
5	99.92%	11	1	5.02%	124.38	155.63	147.56	204.50	0.90	≈ 0
			2	11.72%	157.05	227.16	193.07	252.01	0.91	≈ 0
			3	56.01%	123.80	182.80	143.88	200.72	0.87	≈ 0
			4	10.86%	117.85	333.58	165.29	355.82	0.94	≈ 0
			5	9.39%	144.61	183.42	171.50	220.41	0.88	≈ 0

5. Conclusion

In this paper, a novel 4D trajectory prediction approach that combines clustering model with prediction model is proposed, implemented and simulated for ETA prediction. The proposed model contains clustering-based preprocessing step and MCNN-based machine learning prediction step.

First of all, four steps in the preprocessing part are discussed in detail, including data cleaning and formatting, dimensionality augmentation, PCA and clustering via DBSCAN. They support to cluster different traffic flows into different traffic patterns. Then, MCNN-based prediction model is introduces. Model validation methods are presented to well train the associated prediction model in corresponding clusters.

Table 3: The performance of ETA prediction on NN and MLR with K-means++-based preprocessing step on each outer cross validation fold

Outer CV#	Classification tree		Cluster#	Percentage	K-means++ P. + NN		K-means++ P. + MLR			
	Accuracy	Size			MAE[s]	RMSE[s]	MAE[s]	RMSE[s]	R^2	p-value
1	96.57%	189	1	56.58%	132.95	181.09	149.10	205.64	0.86	≈ 0
			2	13.00%	182.60	264.28	216.32	297.36	0.88	≈ 0
			3	18.21%	119.65	189.74	124.61	173.45	0.88	≈ 0
			4	12.20%	167.83	248.05	201.59	279.07	0.88	≈ 0
2	98.61%	121	1	58.07%	141.21	235.99	158.17	246.24	0.86	≈ 0
			2	11.18%	184.53	345.09	217.16	348.63	0.92	≈ 0
			3	17.85%	128.03	186.80	133.91	191.13	0.90	≈ 0
			4	12.90%	187.92	286.01	217.12	306.47	0.89	≈ 0
3	95.85%	225	1	57.99%	141.89	209.23	166.78	233.82	0.86	≈ 0
			2	11.98%	164.78	210.26	250.49	301.44	0.88	≈ 0
			3	17.13%	102.30	133.47	127.53	160.14	0.87	≈ 0
			4	12.90%	157.33	209.05	201.26	261.95	0.88	≈ 0
4	96.79%	203	1	58.36%	118.65	162.36	141.73	187.14	0.85	≈ 0
			2	12.09%	161.72	214.28	198.10	268.12	0.89	≈ 0
			3	12.80%	115.25	146.73	127.20	164.58	0.86	≈ 0
			4	16.75%	131.08	230.94	137.81	242.00	0.87	≈ 0
5	96.06%	195	1	45.36%	125.98	191.33	138.49	201.65	0.86	≈ 0
			2	17.28%	135.32	232.01	140.62	226.50	0.88	≈ 0
			3	24.68%	150.65	243.84	157.07	257.20	0.88	≈ 0
			4	12.68%	162.09	226.63	190.85	253.26	0.88	≈ 0

Table 4: The average performance of ETA prediction on NN and MLR with/without preprocessing step over 5 folds

Preprocessing	Classification tree		NN		MLR			
	Accuracy	Avg number of nodes	MAE[s]	RMSE[s]	MAE[s]	RMSE[s]	R^2	p-value
Without	-	-	145.58	218.26	222.88	302.59	0.77	≈ 0
K-means++	96.78%	187	139.09	212.20	160.01	232.76	0.81	≈ 0
DBSCAN	99.81%	17	135.62	210.21	162.85	236.04	0.81	≈ 0

Numerical experiments demonstrate that NN with DBSCAN preprocessing performs best in terms of MAE and RMSE, compared with other 5 methods, including NN with K-Means preprocessing, NN without preprocessing, MLR without preprocessing, MLR with K-Means preprocessing and MLR with DBSCAN preprocessing. Clearly, the hybrid machine learning prediction model MCNN with DBSCAN preprocessing can make an accurate, efficient and robust short term trajectory prediction in TMA.

Future work could be conducted on a larger time scale or special scale. Moreover, limited by the size of dataset that we obtain at this moment, deep learning models are not used in this paper. More advanced deep learning model can be applied to improve the prediction performance when the dataset is enough for training the model.

6. Acknowledgement

The authors would like to thank Serge Roux who provided us with the data used in the computational experiments. Special thanks also go to colleagues in the Optim group of ENAC for their generous suggestions.

References

Alligier, R., Gianazza, D., Durand, N., 2015. Machine learning applied to airspeed prediction during climb, in: ATM seminar 2015, 11th USA/EUROPE Air Traffic Management R&D Seminar.

Arthur, D., Vassilvitskii, S., 2007. k-means++: The advantages of careful seeding, in: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics. pp. 1027–1035.

Ayhan, S., Samet, H., 2015. Diclerge: Divide-cluster-merge framework for clustering aircraft trajectories. Proceedings of the 8th ACM SIGSPATIAL IWCTS, Seattle, WA .

CAAC Operation Monitoring Center, 2018. Report on National Civil Aviation Flight Operation Efficiency in 2017 (Chinese). Technical Report.

Chen, J., Wang, R., Liu, L., Song, J., 2011. Clustering of trajectories based on hausdorff distance, in: Electronics, Communications and Control (ICECC), 2011 International Conference on, IEEE. pp. 1940–1944.

Eckstein, A., 2009. Automated flight track taxonomy for measuring benefits from performance based navigation, in: Integrated Communications, Navigation and Surveillance Conference, 2009. ICNS'09., IEEE. pp. 1–12.

Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise., in: Kdd, pp. 226–231.

FAA, E., 2010. Common trajectory prediction structure and terminology in support of SESAR & NextGen. Technical Report. USA: FAA/EUROCONTROL AP 16.

Gariel, M., Srivastava, A.N., Feron, E., 2011. Trajectory clustering and an application to airspace monitoring. IEEE Transactions on Intelligent Transportation Systems 12, 1511–1524.

Guan, X., Zhang, X., Han, D., Zhu, Y., Lv, J., Su, J., 2014. A strategic flight conflict avoidance approach based on a memetic algorithm. Chinese Journal of Aeronautics 27, 93–101.

Hong, S., Lee, K., 2015. Trajectory prediction for vectored area navigation arrivals. Journal of Aerospace Information Systems .

Hrastovec, M., Solina, F., 2016. Prediction of aircraft performances based on data collected by air traffic control centers. Transportation Research Part C: Emerging Technologies 73, 167–182.

Kaneshige, J., Benavides, J., Sharma, S., Martin, L., Panda, R., Steglinski, M., 2014. Implementation of a trajectory prediction function for trajectory based operations, in: AIAA Aviation Atmospheric Flight Mechanics Conference, No. AIAA.

Le Fablec, Y., Alliot, J.M., 1999. Using neural networks to predict aircraft trajectories., in: IC-AI, pp. 524–529.

Leege, A.D., Paassen, M.V., Mulder, M., 2013. A machine learning approach to trajectory prediction, in: Aiaa Guidance, Navigation, and Control.

MacQueen, J., et al., 1967. Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Oakland, CA, USA.. pp. 281–297.

Nicol, F., Puechmorel, S., 2016. Unsupervised aircraft trajectories clustering: a minimum entropy approach. ALLDATA 2016 , 44.

Nuic, A., Poles, D., Mouillet, V., 2010. Bada: An advanced aircraft performance model for present and future atm systems. International Journal of Adaptive Control and Signal Processing 24, 850–866.

Porretta, M., Dupuy, M.D., Schuster, W., Majumdar, A., Ochieng, W., 2008. Performance evaluation of a novel 4d trajectory prediction model for civil aircraft. The Journal of Navigation 61, 393–420.

Reynaldi, A., Lukas, S., Margaretha, H., 2012. Backpropagation and levenberg-marquardt algorithm for training finite element neural network, in: Computer Modeling and Simulation (EMS), 2012 Sixth UKSim/AMSS European Symposium on, IEEE. pp. 89–94.

Rousseeuw, P.J., 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics 20, 53–65.

Runkler, T.A., 2012. Data Analytics. Springer.

Sander, J., Ester, M., Kriegel, H.P., Xu, X., 1998. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. Data mining and knowledge discovery 2, 169–194.

Tang, X., Gu, J., Shen, Z., Chen, P., 2015. A flight profile clustering method combining twed with k-means algorithm for 4d trajectory prediction, in: Integrated Communication, Navigation, and Surveillance Conference (ICNS), 2015, IEEE. pp. S3–1.

Tastambekov, K., Puechmorel, S., Delahaye, D., Rabut, C., 2014. Aircraft trajectory forecasting using local functional regression in sobolev space. Transportation research part C: emerging technologies 39, 1–22.

Trivedi, S., Pardos, Z.A., Heffernan, N.T., 2015. The utility of clustering in prediction tasks. arXiv preprint arXiv:1509.06163 .

Wang, Z., Liang, M., Delahaye, D., 2017. Short-term 4D Trajectory Prediction Using Machine Learning Methods, in: SID 2017, 7th SESAR Innovation Days, Belgrade, Serbia. URL: https://hal-enac.archives-ouvertes.fr/hal-01652041.

Xi, L., Jun, Z., Yanbo, Z., Wei, L., 2008. Simulation study of algorithms for aircraft trajectory prediction based on ads-b technology, in: System Simulation and Scientific Computing, 2008. ICSC 2008. Asia Simulation Conference-7th International Conference on, IEEE. pp. 322–327.