Fourteenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2021)

Early Detection of Night Curfew Infringements by Delay Propagation with Neural Networks

Ramon Dalmau & Giuseppe Murgese
Network Research Unit (NET)
EUROCONTROL
Brétigny-Sur-Orge, France

Yves De Wandeler & Ricardo Correira
Central Office for Delay Analysis (CODA)
EUROCONTROL
Brussels, Belgium

Alan Marsden
Airports Research Unit (APT)
EUROCONTROL
Brétigny-Sur-Orge, France

***Abstract*—Airport night curfews are restrictions applied at some airports that prohibit operations during certain hours of the night, aiming to reduce noise nuisance in the surrounding neighborhood. Despite effectively reducing noise exposure for local residents, these environmental measures can have negative economic and operational effects on the airspace user and the airport, as well as a negative experience for the passenger. This paper presents a model that, for each flight and well before the starting time of the curfew period, is able to provide the probability (risk) of night curfew infringement. The risk of night curfew infringement is computed from the start time of the restriction and the distribution of in-block times. The former is known for each airport, while the latter is provided by a neural network which was trained on historical data to predict the propagation of arrival delay along the sequence of flights of an aircraft. Results show that the model significantly improves the in-block time predictions, if compared to the current solution. Furthermore, the risk indicator could assist in identifying flights with potential risk of night curfew infringements within a reasonable time frame to implement effective mitigation actions.**

***Keywords*—delay propagation; machine learning; curfew**

I. Introduction

Aircraft noise is a major concern for millions of people living in the vicinity of airports. Despite the continuous efforts of the industry to design quieter engines and airframes, several airports have taken operational measures to address the issue.

Night curfews are environmental restrictions imposed at some airports, which prohibit take-offs and/or landings during certain night-time hours in order to mitigate the noise nuisance on the local airport community. These night curfews could be *partial* or *total*. At Paris-Orly, for instance, a total night curfew from 23:30 to 06:00 (local time) has been implemented since 1968. Berne-Belp is another example of airport with total night curfew, where operations are not permitted between 23:00 and 06:00 (local time). In some airports, like Rome–Fiumicino, a partial night curfew is applied only to the runway that is closer to populated areas. Liverpool and Stockholm-Arlanda are examples of airports implementing a partial night curfew, which restricted period time depends on the day of the week [1], [2]

Night curfews are common across Europe, and although they have positive effects for the surrounding residents, they could also cause significant operational and financial disruptions. In some cases, a flight running late may have to divert to an alternate airport if it is unable to reach the airport before the start of the night curfew, and therefore passengers would have to be shuttled between airports. In other cases, a flights may be cancelled if it is certainly known that it would infringe the night curfew at the destination airport, or not be able to depart from the origin before the end of the restriction.

In this context, developing effective methods to accurately identify flights with potential risk of night curfew infringement within a reasonable time frame is of paramount importance to apply early responses and, consequently, to mitigate the negative economic and operational performance impact for the airspace user and the airport, as well as the unpleasant experience for the passengers.

Night curfews infringements are often caused by a delay propagation along the sequence of flights *legs* scheduled for an aircraft during the course of the day. The delay that propagates to a flight depends on the length of scheduled turn-around time, the arrival punctuality of the previous flight and the operational efficiency of the ground-handling services at the airport, among other factors. For instance, an aircraft with a very tight schedule that is delayed in its very first flight could propagate such delay throughout the consecutive flights. If this were the case, it could not be able to operate the last flight of the day because of the night curfew at the origin or destination airport. In other situations, however, the flight could be able to absorb the delay during the course of the day, or the airspace user may decide to swap the aircraft and *deflect* the delay to an airport with no curfew.

Unfortunately, it is not straightforward to anticipate whether a flight will infringe the night curfew or not, especially when the assessment is performed several hours (and flight legs) in advance, when the uncertainty of the actual delay propagation along the scheduled sequence is very high.

This paper presents a two-stages model, which is capable to detect flights with potential risk of night curfew infringement well before the starting time of the restricted period. Instead of relying on hard-coded rules laboriously designed by domain experts, the model has been trained on historical traffic data by using state-of-the-art machine learning techniques.

The model performs predictions per aircraft (i.e., tail number), and can be queried at any time during the day using the most recent information for each one of the flights that the aircraft has already operated (if any) and those planned until the end of the journey. During the first stage, the model predicts the propagation of arrival delay along the sequence

of subsequent flights. The result is a distribution of predicted in-block times for the very last flight of the sequence. During the second stage, the distribution of predicted in-block times is used to analytically compute the probability of night curfew infringement, given the known start time of the restriction.

The input features used to train the model have been obtained by combining two sources of data: Enhanced Tactical Flow Management System (ETFMS) flight data (EFD) messages, and the published airline schedules. The model has been trained on sequences of flights (each one corresponding to the state of the scheduled sequence of flights for a particular aircraft at a given time) from 2019. The ability of the model to generalise has been verified by assessing its performance on *unseen* sequences from 2018. Last but not least, a validation exercise has been also performed for the COVID-19 period, aiming to measure the hypothetical performance degradation in such exceptional and unfortunate traffic situation.

II. Background

Understanding the factors that drive delay propagation across the flights of the air traffic network has attracted many researchers in the recent years. This work attempts to address this problem from a data-driven point of view, taking advantage of the large amount of data available and the outstanding predictive performance of neural networks. Section II-A reviews previous works on the modelling and prediction of delay propagation. Section II-B describes the working principle of recurrent neural networks, an architecture designed to make predictions from sequential data.

A. Delay propagation

Some authors analysed the intensification and smoothing factors of this widespread phenomenon. For instance, Ref. [3] investigated the relationship between the planned schedules of aircraft and crew, and the potential for delays to propagate across the network. Results showed that the planned schedules are often very tight, thus limiting the ability to absorb delays.

In parallel, Ref. [4] introduced several indicators to quantify the propagation of delays, as well as to better understand the amplifying and mitigating factors. Interesting results showed that almost half of the departure delays in Europe are caused by reactionary delays. Furthermore, empirical results suggested that, the higher the inbound delay and the lower absorbed delay, the more reactionary delay is propagated to the subsequent flight legs. The analysis also revealed significant differences in techniques used to mitigate reactionary delay, depending on the business strategy of the airline.

Later on, Ref. [5] proposed an analytical model to quantify propagated and newly formed delays along the sequence of flights that an aircraft operates during the day. The model was formulated by considering that airlines (deliberately) insert buffer into the flight schedules and ground turn-around operations, which could be used to absorb the delay. In addition, the influence of various factors involved on the initiation and progression of the reactionary delays were studied by using a novel discrete-continuous econometric model.

Other relevant works on this field modelled the propagation of delays at network-wide level by using Bayesian networks [6], causality networks based on the Granger causality test [7], queuing and network decomposition methods [8], as well as data-driven models [9] and simulators [10].

In the recent years, the use of historical data and machine learning has shown potential to improve the accuracy of delay predictions. For example, Ref. [11] assessed the strategic flight schedules (i.e., slots) of an airport by using machine learning to predict potential flight delays and cancellations. Another example is Ref. [12], which recently proposed a causal machine learning algorithm to predict the probability (risk) of flight delays and to identify the driving features.

In a previous work [13], an explainable machine learning model built on gradient boosted decision trees was used to improve the predictability of take-off times. A comprehensive analysis of features importance showed that, when the time available to perform the turn-around is below a given threshold (which in turn depends on the airline, airport and aircraft type, among other factors), the impact of this input feature on the prediction rapidly increases. These results suggested that data-driven models could capture reactionary delays caused by delay aggregation and propagation throughout the consecutive flight legs of a given aircraft during the day.

The architecture of the neural network proposed herein, as well as the input features that the model uses to predict the delay propagation as required to assess the risk of night curfew infringement, were inspired by the amplifying or mitigating factors identified in the aforementioned works.

Different from previous works, this paper proposes a probabilistic model, which does not only provide the expected value of the delay, but also the uncertainty associated to the prediction (used to compute the risk). Last but not least, the model was trained and validated on real flight data, and was explicitly designed to work in a real-time operational environment, taking advantage of the most up-to-date information.

B. Recurrent neural networks

In this paper, vectors are denoted with bold fonts, e.g., $\boldsymbol{a}$, while matrices use bold computer modern calligraphy fonts, e.g., $\boldsymbol{\mathcal{A}}$. Furthermore, the following notation is used when referring to sequences $(\boldsymbol{a}_t)_{t=1}^{T} = (\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_T)$.

Since their (presumably) first appearance in Ref. [14], recurrent neural networks (RNN) rapidly became very popular in many natural language processing (NLP) applications due to their outstanding ability to capture the underlying sequential structure present in the language.

Roughly speaking, RNN have a sort of *memory* over previous computations, and use this information when processing the current input of the sequence. This information is stored in the hidden state vector $\boldsymbol{h}_t \in \Re^{n_h}$. At each time step, the RNN uses the current inputs vector $\boldsymbol{x}_t \in \Re^{n_x}$ to update the hidden state. For vanilla RNN, the update function is:

$$\boldsymbol{h}_t = \tanh\left(\boldsymbol{\mathcal{W}}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{\mathcal{W}}_{hx}\boldsymbol{x}_t\right), \tag{1}$$

where $\boldsymbol{\mathcal{W}}_{hh} \in \Re^{n_h \times n_h}$ and $\boldsymbol{\mathcal{W}}_{hx} \in \Re^{n_h \times n_x}$ are matrices of parameters to be learned during the training process[1].

[1]Throughout this paper, if not explicitly mentioned, the bias vector is omitted for the sake of simplicity.

Note that, at the very first time step, the hidden state depends on the first inputs vector $\boldsymbol{x}_1$ as well as the initial hidden state $\boldsymbol{h}_0$. The default approach consists of initialising the hidden state with zeros (i.e., $\boldsymbol{h}_0 = \boldsymbol{0}$). Another alternative is to consider $\boldsymbol{h}_0$ as additional trainable parameters. The most reasonable approach, however, consists of conditioning $\boldsymbol{h}_0$ on static inputs (i.e., those not changing along the sequence).

The vanilla RNN update described by Eq. (1), however, frequently leads to the problem of vanishing gradients, which negatively influences the learning of long sequences. To solve this problem, a popular way is to use LSTM (long short-term memory) or gated recurrent unit (GRU) [15]. In this paper, the latter architecture has been selected because it is computationally faster than LSTM, and provides similar or even better performance in a wide variety of applications.

In this context, let us define GRU as the neural network that, given a sequence of inputs $(\boldsymbol{x}_t)_{t=1}^T$ and the initial hidden state $\boldsymbol{h}_0$, generates a sequence of hidden states $(\boldsymbol{h}_t)_{t=1}^T$:

$$(\boldsymbol{h}_t)_{t=1}^T = \text{GRU}\left((\boldsymbol{x}_t)_{t=1}^T, \boldsymbol{h}_0\right). \tag{2}$$

At every step $t = 1, 2, \ldots, T$ of the input sequence, the following operations are performed to update the hidden state:

$$\begin{aligned}
\boldsymbol{z}_t &= \sigma\left(\boldsymbol{\mathcal{W}}_{zh}\boldsymbol{h}_{t-1} + \boldsymbol{\mathcal{W}}_{zx}\boldsymbol{x}_t\right), \\
\boldsymbol{r}_t &= \sigma\left(\boldsymbol{\mathcal{W}}_{rh}\boldsymbol{h}_{t-1} + \boldsymbol{\mathcal{W}}_{rx}\boldsymbol{x}_t\right), \\
\boldsymbol{n}_t &= \tanh\left(\boldsymbol{\mathcal{W}}_{nx}\boldsymbol{x}_t + \boldsymbol{r}_t \odot \boldsymbol{\mathcal{W}}_{nh}\boldsymbol{h}_{t-1}\right), \\
\boldsymbol{h}_t &= (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \boldsymbol{n}_t,
\end{aligned} \tag{3}$$

where the *update* gate $\boldsymbol{z}_t \in \Re^h$ controls the extent to which the information of the previous state is transferred into the current state, and the *reset* gate $\boldsymbol{r}_t \in \Re^h$ controls how much the previous state contributes to the *new* candidate state $\boldsymbol{n}_t \in \Re^h$. In Eq. (3) and for the remainder of the paper, σ represents the sigmoid function, and $\odot$ the Hadamard product.

This architecture effectively allows for control of the flow of hidden state along the sequence, determining how much hidden state is propagated to the next step of the sequence, when to reset the hidden state, and how to update the hidden state based on the current inputs. Note that, replacing *hidden state* by *distribution of delay* and *inputs* by *flight attributes* in the previous sentence, one could already guess the working principle of the model proposed in the following section.

For many tasks, it is beneficial to have access to past as well as future information (if available). The basic GRU architecture described by Eq. (3), however, processes sequences only in the forward direction. This implies that $\boldsymbol{h}_t$ is an implicit function of $(\boldsymbol{x}_q)_{q=1}^{q<t}$. The bidirectional GRU (BiGRU) extends the unidirectional case by introducing a second GRU (with different trainable parameters), which processes the input sequence in the opposite direction. Then, the hidden state at each time step is the result of concatenating the hidden states of the forward ($\overrightarrow{\boldsymbol{h}_t}$) and the backward ($\overleftarrow{\boldsymbol{h}_t}$) GRUs:

$$\boldsymbol{h}_t = \overrightarrow{\boldsymbol{h}_t} \parallel \overleftarrow{\boldsymbol{h}_t}. \tag{4}$$

III. Mathematical Model

Let us consider an aircraft that will be used to perform a sequence of T flights along the day, ordered by off-block time. At any time during the day, each flight t of the sequence, with $t = 1, 2, \ldots, T$, can be described by a vector of features $\boldsymbol{x}_t$ that represent its most up-to-date state. Furthermore, the aircraft has some static characteristics $\boldsymbol{c}$ that are identical to all flights in the sequence, such as the aircraft type.

The main component of the model consists of a BiGRU that generates a sequence of hidden states $(\boldsymbol{h}_t)_{t=1}^T$ out of the sequence of flight features $(\boldsymbol{x}_t)_{t=1}^T$, generates a sequence of hidden states. The initial hidden states of the forward and and backward GRUs are conditioned on $\boldsymbol{c}$ as follows:

$$\begin{aligned}
\overrightarrow{\boldsymbol{h}}_0 &= \text{FFNN}\left(\boldsymbol{c}\right), \\
\overleftarrow{\boldsymbol{h}}_0 &= \text{FFNN}\left(\boldsymbol{c}\right).
\end{aligned} \tag{5}$$

A feed-forward neural network (FFNN) could be composed by one or several stacked layers. Each one of these layers is defined by the number of neurons n and the activation function ϕ, and generates a vector of outputs $\boldsymbol{y} \in \Re^n$ by applying the following operation to the inputs $\boldsymbol{x} \in \Re^{|\boldsymbol{x}|}$:

$$\boldsymbol{y} = \phi\left(\boldsymbol{\mathcal{W}}\boldsymbol{x} + \boldsymbol{b}\right), \tag{6}$$

where $\boldsymbol{\mathcal{W}} \in \Re^{n \times |\boldsymbol{x}|}$ is a weighting matrix, and $\boldsymbol{b} \in \Re^n$ is the bias vector. Both $\boldsymbol{\mathcal{W}}$ and $\boldsymbol{b}$ are parameters to be learned. Note that (1) the outputs of the first layer are the inputs of the second layer, and so on; (2) each layer may have a different number of neurons and activation function; and (3) the parameters of the FFNN that generates $\overrightarrow{\boldsymbol{h}}_0$ are different from those of the FFNN that generates $\overleftarrow{\boldsymbol{h}}_0$. The hyper-parameters (number of layers, neurons, activation functions, etc.) of all the components (BiGRU, FFNNs, etc.) of this generic model will be particularised in Section V-A.

Provided that the predictive power of the model (mainly expressed by the number of parameters) and the number of examples processed during training are high enough, the model shall learn to capture in $(\boldsymbol{h}_t)_{t=1}^T$ all the information contained in the input sequence $(\boldsymbol{x}_t)_{t=1}^T$ that could be useful to predict the probability distribution of arrival delay for each non-terminated flight in the sequence of the aircraft.

When training a model that predicts a probability distribution, one must first know which type of distribution best fits the target. Whatever type of distribution is selected, it must be expressed as a parametric function, which parameters determine the exact shape of the distribution. These parameters are the values to be predicted by the model. In this work, it is assumed that the arrival delay can be approximated by a Normal distribution[2]. Accordingly, the model is trained to predict the expected value (μ) and the standard deviation (σ). Given a hidden state vector $\boldsymbol{h}_t$, the model predicts the probability distribution of the corresponding flight t.

[2]Actually, the arrival delay could be better approximated by a log-normal or other distributions with some positive skewness. In this paper, a Normal distribution was used because it is easier to interpret and approximates the arrival delay well enough for the objective of the project.

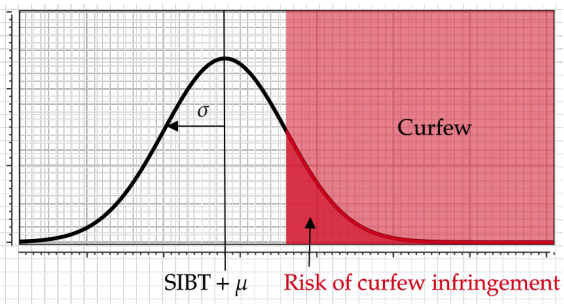


Figure 1: Risk of curfew infringement computation

$$\begin{aligned} \sigma_t &= \text{FFNN}\left(\boldsymbol{h}_t\right) + \epsilon \\ \mu_t &= \text{FFNN}\left(\boldsymbol{h}_t\right) \end{aligned} \tag{7}$$

Note that the support of the standard deviation is $\sigma \in (0, \infty)$. In order to ensure that the model provides a valid prediction, the $\text{ReLU}\left(x\right) = \max\left(0, x\right)$ is used as the activation function for the last layer of the FFNN that generates σ, and a small constant value ϵ is added to the result.

Finally, the parameters of the model described above are optimised to minimise the negative log-likelihood. As a result, the contribution of each sequence of flights to the loss (i.e., penalty) function is given by the following expression:

$$L = -\sum_{t=1}^{T} \log \mathcal{N}\left(y_t \,|\, \sigma_t^2, \mu_t\right) \left(1 - \mathbb{1}_{\{\text{TE}\}}\left(s_t\right)\right), \tag{8}$$

where the indicator function $\mathbb{1}_{\mathcal{E}}$ of the event $\mathcal{E}$ is:

$$\mathbb{1}_{\mathcal{E}}(x) := \begin{cases} 1 & \text{if } x \in \mathcal{E} \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

and y_t is the actual (i.e., true) delay of flight t, computed as the difference between the actual in-block time (AIBT) and the scheduled in-block time (SIBT)[3].

In Eq. (8), s_t represents the state of the flight t at the moment of performing the prediction and TE is the state that identifies terminated flights. Note that terminated flights are not considered in the loss function (predicting the delay of a terminated flight is not interesting), yet they are essential to to predict the delay of the subsequent flights in the sequence.

Once the probability distribution of arrival delays has been computed by the model, one is ready to assess the risk of curfew infringement. First, the distribution is added to the SIBT, in order to obtain the distribution of in-block times. By knowing this distribution and the night curfew period, the probability of arriving after its starting time can be computed analytically. This procedure is illustrated in Fig. 1.

[3]The off-block time is defined as the time that an aircraft pushes back the parking position. Analogously, the in-block time is when the parking brakes have been engaged at the parking position.

IV. Features

At a given time, the vector of features $\boldsymbol{x}_t$ for each flight in a sequence are computed by combining two sources of data: Enhanced Tactical Flow Management System (ETMS) flight data (EFD) messages, and the schedules of the airline.

The ETFMS distributes EFD messages to inform users with the latest updates of the state of a flight. The first event at which a message is sent corresponds to the moment of the flight plan creation, and the distribution stops when the flight terminates. An EFD message includes static attributes of the flight (such as departure and destination airports, aircraft type and registration, etc.) but also dynamic features that may change with time. In between the transmission of the first and the last messages, EFD are distributed whenever one of the fields changes, and also at regular time intervals.

Prior to off-block, the EFD messages provide accurate information about the estimated off-block time (EOBT), the estimated time of arrival (ETA), the expected taxi time, the aircraft type and registration number, as well as the air traffic flow management (ATFM) delay and the list of regulations to which the flight is subject (if any). The ETFMS also knows actual and predicted information after off-block, including the actual off-block time (AOBT) and the most up-to-date airborne position. The actual time of arrival (ATA) is also known by the ETFMS once the flight terminates. Further details of the ETFMS, the information included in the EFD messages, as well as the list of events that could trigger the distribution of EFD messages can be found in [16].

The schedules of the airline correspond to the airport slot times, i.e., the scheduled off-block time (SOBT) and SIBT. Note that these times are static per flight. Furthermore, some flights may not be scheduled. In these cases, the EOBT and ETA (plus the average taxi-in time at destination airport) from the flight plan are used instead of SOBT and SIBT, respectively. In addition, non-scheduled flights are *flagged* with a boolean indicator, in order to inform the model.

Table I lists the input features for each flight in the sequence planned by an aircraft, $\boldsymbol{x}_t$, as observed at a given time during the day. Table II lists the static features related to the whole sequence at the same time instant, $\boldsymbol{c}$, which are used to condition the hidden state of the BiGRU (see Eq. (5)).

Table I includes basic information of the flight, such as the departure and arrival airports, the airline, etc., information about the most penalising ATFM regulation affecting the flight and the associated ATFM delay (if any), the latest event that triggered an EFD message and the state of the flight at that moment, as well as some features computed by subtracting timestamps. In Table I, OBT and IBT represent the most accurate off-block and in-block times when performing the prediction, respectively. That is, the OBT is assigned to the first available milestone by order of priority: (1) AOBT if the flight has already departed, (2) TSAT if the flight has been included in the ATC pre-departure sequence of a collaborative decision making (CDM) airport, (3) COBT if the flight is regulated, (4) TOBT if a departure planning information (DPI) message has been received from a CDM airport, and (5) EOBT otherwise. A similar criteria applies for the IBT.

TABLE I. List of features for each flight in a sequence at a given time (i.e., $\boldsymbol{x}_t$)

Feature	Type of transformation
Departure airport (ICAO code) Arrival airport (ICAO code) Aircraft operator Airline Traffic volume associated to the regulation[2][3] Geographical entity associated to the traffic volume[2] Type of geographical entity[2] Reason of the regulation[2] Type of event that triggered the EFD message (see Ref. [16] for a detailed description) Class of the event that triggered the EFD message (see Ref. [16] for a detailed description) ATFM state of the flight (see Ref. [16] for a detailed description) Ready state (see Ref. [16] for a detailed description) Flight model type (see Ref. [16] for a detailed description) Type of departure airport in CDM, advanced air traffic control (ATC) tower, or conventional Collaborative decision making (CDM) state[1] (see Ref. [16] for a detailed description)	Embedding
Estimated taxi-out time at departure airport Average taxi-in time at destination airport Time to SOBT Time to SIBT Time from SOBT to OBT (Current departure delay) Time from SIBT to IBT (Current arrival delay) Time from SOBT to EOBT Time from IOBT (initial off-block time) to EOBT Time from TOBT (target off-block time) to TSAT (target start-up approval time) [1] Time from SOBT to SIBT (Scheduled block time) Time from OBT to IBT (Current block time) Difference between current and scheduled block times Time from SIBT of the previous flight leg to SOBT (Scheduled turn-around time) Time from IBT of the previous flight leg to OBT (Current turn-around time) Difference between current and scheduled turn-around times ATFM delay[2] Time since last EFD message Time since first EFD message (i.e., moment of the flight plan creation)	Power (Yeo-Johnson) $\rightarrow$ Standarisation
IOBT missing indicator TSAT missing indicator SOBT missing indicator	Boolean missing indicator
Hour of SOBT	Cyclic
Relative flight leg in the sequence (0 being the first, 1 being the last)	-

TABLE II. List of features for a sequence at a given time (i.e., $\boldsymbol{c}$)

Feature	Type of transformation
Tail number (i.e., registration) of the aircraft Aircraft type	Embedding
Day of the week Month of the year Hour of the day	Cyclic
Length of the sequence divided by 10	-

Some of the features are discrete (i.e., categorical), such as the airline, while some others are continuous (i.e., numerical), like the ATFM delay. Some machine learning modes, like decision trees, are robust to arbitrary scaling of the numerical features and only require categorical features to be encoded as integers. Neural networks, however, prefer numerical features to be standardised and/or normalised, and require special attention when dealing with categorical features.

[1]Present in flights departing from CDM and advanced ATC tower airports.

[2]Present in regulated flights.

[3]A traffic volume is related to a single geographical entity (either an aerodrome, a set of aerodromes, an airspace or a point), and may consider all traffic passing through that entity or only specific flows.

On the one hand, normalisation consists of re-scaling a feature from the original range to a smaller scale, typically through dividing by the maximum absolute value or performing a min-max transformation. This conversion, however, is not appropriate for features whose distribution exhibits outliers or skewness. On the one hand, standardisation consists of subtracting the mean and then dividing the result by the standard deviation, under the assumption that the feature is normally distributed. Very often, however, a feature presents severe skewness, and therefore the assumption of normality does not hold. In such cases, power transformations typically help to make the distribution of a feature more Gaussian-like, stabilising the variance and reducing the skewness.

Simple power transformations, such as calculating the logarithm or the square root may not always help to remove the skewness. In this paper the parametric Yeo-Johnson transformation has been applied to some numerical features, followed by a standarisation [17].

Regarding categorical features, one-hot encoding is a popular method for converting them into numerical features. Unfortunately, one-hot encoding of high-cardinality features (such as the departure airport) often results in an unnecessary amount of computational resources. An alternative to one-hot encoding are the embeddings layers. An embedding automatically learns the representation of each category of a feature in a multi-dimensional space [18]. The result of the embedding is a vector of continuous values for each category, which values are closer for categories with similar effect to the task that the neural network aims to accomplish.

Other features, especially those related with time (e.g., hour of the day, day of the week), are cyclical in nature. A common method for encoding cyclical features is to map the data into two dimensions, using the sine and consine transformations. The boolean missing indicators only inform about the presence or absence of certain features. Finally, a small set of features can be fed directly to the neural network, without prior (or just a very simple) transformation.

Let us define $\boldsymbol{x}_{t,e}$ as the concatenation of the embedding vectors of all categorical features of the flight t of the sequence. Let us also define $\boldsymbol{x}_{t,n}$ as the vector of numerical features (which include those transformed with Yeo-Johnson algorithm, cyclic features, the boolean missing indicators, and features not requiring a transformation). Then, the vector of features for the flight t of the sequence is $\boldsymbol{x}_t = \boldsymbol{x}_{t,e} \parallel \boldsymbol{x}_{t,n}$. The length of $\boldsymbol{x}_{t,n}$ is 24 (note that the hour of the SOBT is mapped to two features), while the $\boldsymbol{x}_{t,e}$ is the sum of the embedding size of all categorical features. Assuming and embedding size of 32, for instance, and having 15 categorical variables, then $|\boldsymbol{x}_{t,e}| = 480$. In practice, however, each categorical feature typically has its specific embedding size, which is an additional hyper-parameter to be selected. Some models, like the one presented herein, include a large number of categorical variables, making the fine-tuning of the embedding sizes impractical. In this case, a simple rule of thumb is to set the embedding size to $\min(n/2, 50)$, where n is the number of categories of the categorical feature. Note that the maximum and minimum embedding sizes could be also considered as additional hyper-parameters of the model.

Similarly, $\boldsymbol{c} = \boldsymbol{c}_e \parallel \boldsymbol{c}_n$ is the condition vector of an aircraft, being $\boldsymbol{c}_e$ the vector of embeddings and $\boldsymbol{c}_n$ the vector of numerical features. In this case, the length of $\boldsymbol{c}_n$ is 7.

V. Results

The model presented in Section III, which uses the features listed in Section IV to predict the distribution of arrival delay, has been trained on real flight data. Section V-A describes the setup of the experiment. Section V-B lists the final hyper-parameters of the model. An illustrative example is shown in Section V-C. Finally, the performance metrics as a result of the validation exercise are discussed in Section V-D.

A. Setup of the experiment

Each example shown to the model during training consists of the most up-to-date information for the whole sequence of flights that an aircraft operates during the day, inspected at a given time (or *snapshot*). In this paper, an snapshot is taken whenever one of the following events happens for whichever flight in the sequence: (1) submission of the initial flight plan, (2) reception of the first airborne message, (3) termination, (4) suspension of the flight, or (5) ATFM slot allocation (typically 2 hours before EOBT for regulated flights).

Consequently, the same sequence of flights may be shown to the model several times during training, depending on the number of events triggered by these flights, operated by the same aircraft. Each one these examples, however, will represent a very different picture of the same sequence, therefore providing new information for the model to learn. Note that, at a given snapshot, some of the flights may be terminated. These flights are not removed from the sequence, since they could provide priceless information (especially the last flight leg that was operated), yet they are not considered in the loss function (as discussed in Section III).

The training examples have been obtained from 280 random days of 2019 (with 1.8M sequences from 23K aircraft). From the remaining days of 2019, 20 days (with 128K sequences from 15K aircraft) were used for early stopping and hyper-parameters tuning (i.e., the validation set), and the other 60 days (with 390K sequences from 19K aircraft) to assess the performance of the model on unseen data. Two additional validation exercises have been performed. During the first validation exercise, the predictions of the model from June to December 2018 (with 1.4M sequences from 21K aircraft) were compared with the actual delays. The second validation exercise was performed using flight data from January to May 2020 (with 520K sequences from 18K aircraft), including the lock-down period caused by the COVID-19 pandemic with an unusual traffic demand.

B. Hyper-parameters of the model and metrics history

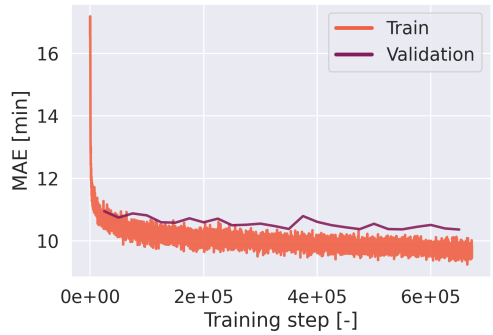
Table III lists the hyper-parameters of the model, which were selected based on manual trial and error. In Table III, the succession of layers for a FFNN is denoted by $\rightarrow$, where the number represents the number of neurons and, if not explicitly specified, the activation function of each layer is the ReLU.

TABLE III. Hyper-parameters of the model

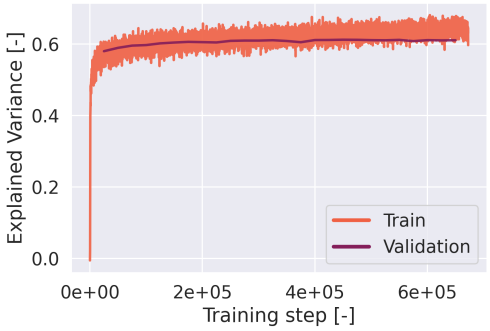
Hyper-parameter	Value
Batch size / Learning rate / Epochs	64 / 1e-4 / 3
Gradient clipping value	5
Min. / Max. embedding size	4 / 64
Hidden state size (n_h)	256
Number of recurrent layers	1
FFNN of Eq. (7) for μ	256→128→64→32→1 (linear)
FFNN of Eq. (7) for σ	256→128→64→32→1
FFNNs of Eq. (5) for $\overrightarrow{\boldsymbol{h}_t}$ and $\overleftarrow{\boldsymbol{h}_t}$	256

Figure 2 shows two metrics that explain the performance of the model for the train and validation sets, as a function the number of training steps. Figure 2(a) shows the evolution of the Mean Absolute Error (MAE) of the predictions.

Figure 2(b) shows the explained variance. The metrics for the validation set have been computed every 25K training steps, aiming to reduce the time required to train the model (6.2 hours using a Nvidia GeForce GTX960 1024MHz 4GB).



(a) Mean absolute arrival delay prediction error



(b) Explained variance of the predicted arrival delay

Figure 2: Training metrics history

Figure 2 shows that the model rapidly converged to a MAE around 10 min and explained variance of 0.6 after updating its 2.5M parameters during 100K training steps (i.e., after processing around 64M of sequences). Then, additional training steps effectively improved the performance metrics on the training set, but the ability for the model to generalise (as described by the validation set metrics) remained stable.

C. Illustrative example

Figure 3 shows an example to illustrate how the model could be used in real-life operations. Each region of the vertical axis shows a different call of the model, ordered in temporal order from the top to the bottom. The bold numbers in the left show the remaining hours until the start of the curfew, and the risk of curfew infringement as predicted by the model. The crosses and the solid dots represent the scheduled and actual in-block times, respectively, for each flight in the sequence. Note that these markers are static and do not change between calls. The vertical dashed lines, however, do change because they represent the most recent predictions of the ETFMS when calling the model. Finally, the distributions are the predictions performed by the model. The horizontal axis represents the relative time from in-block time to the start time of the restriction. Positive values correspond to the curfew period. It should be noted that, once a flight terminates, it is not longer shown for the sake of clarity.

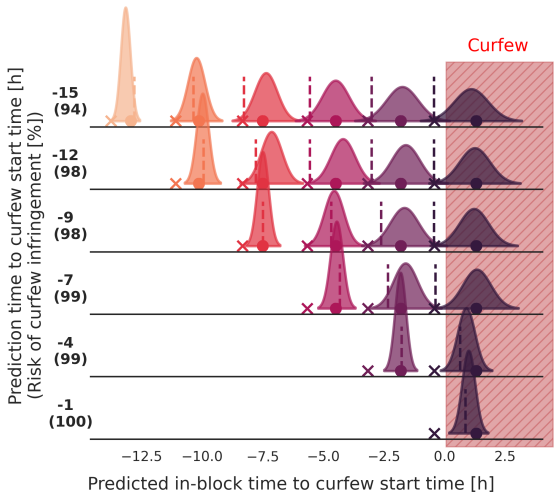


Figure 3: Example (×: SIBT, ---: ETFMS, ●: AIBT)

Figure 3 shows the evolution of the predictions and the actual delays for an aircraft with a sequence of six flights, which last flight was scheduled only 30 min before the night curfew. The first prediction was performed 15 hours before the start time of the restriction, right before the departure of the first flight, which actual delay was around 30 min. Given the short look-ahead time of the prediction, the model was able to provide an accurate and certain prediction for the first leg. Note that, at that time, the predictions of the ETFMS were excellent for the two first flights. The ETFMS, however, missed the delay propagation to the third and subsequent flights, which was the cause of the night curfew infringement. Despite having 6 flights ahead and performing the prediction 15 hours before the start time of the restriction, the proposed model was able to predict the potential delay propagation, resulting in a risk of night curfew infringement of 94%.

As flights were operated and confirmed information was provided to the model to perform the predictions, its confidence about the risk of night curfew infringement increased. It was not until 4 hours before the starting time of the restriction that the ETFMS was able to predict the infringement. At that moment, the the probability of night curfew infringement according to the model was 99%, which strong confidence was supported by a relatively small standard deviation.

D. Performance metrics

Analysing particular examples is not only entertaining, but also gives a clue of the behavior of the model in operational situations. Yet, the performance of a model must be corroborated on a large amount (and variety) of samples in order to provide statistically meaningful figures, and certainly assess the potential benefits that it could provide at large-scale.

1) 60 random days of 2019: Figure 4 shows the MAE of the arrival delay predictions for the last flight of the sequence (i.e., the one for which the curfew infringement is assessed), as a function of the time to its SOBT (in the horizontal axis), and the number of flight legs ahead (in the vertical axis). 0 flights ahead indicates that, at the moment of performing the prediction, the next flight is the last of the sequence.

Clearly, the cells are mostly aligned along the diagonal of the figure, since it is very unlikely to be at 2 hours before the last SOBT with still 6 flight legs ahead, for instance. Cells with a frequency of samples lower than 1% were removed from the figure to facilitate the interpretation of results.

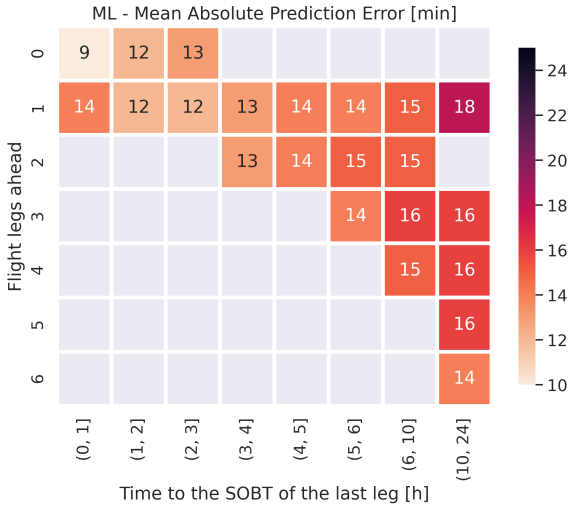


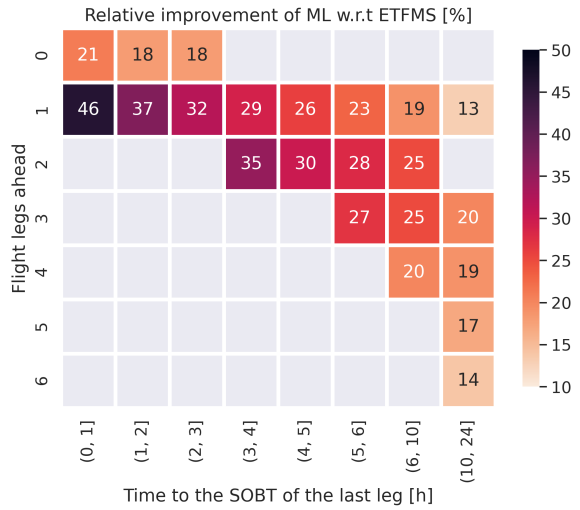
Figure 4: Arrival delay prediction MAE (60 days of 2019)

According to Fig. 4, and as expected, the performance of the model typically improves as the number of flights legs ahead and the time to SOBT decrease. It should be noted, however, that the number of samples in each cell may be different, thus comparing the performance of different cells may not be fair. In any case, the prediction error of the model (in average terms) ranges from 9 min to 18 min.

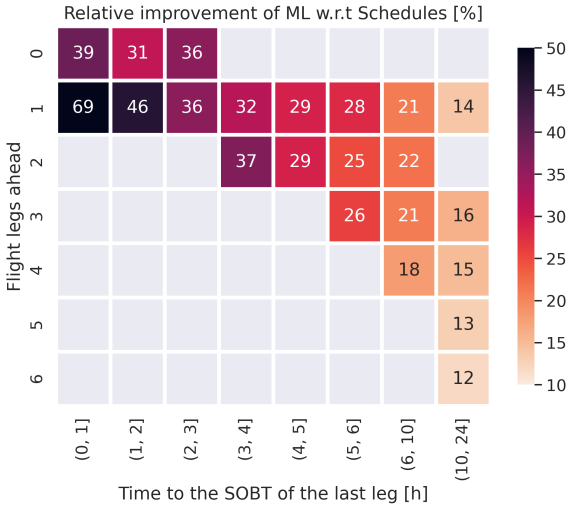
Figure 4 definitely shows the performance of the model in terms of predictive accuracy. Yet, the added value of the model needs to be assessed by comparing these results with the performance of the current predictions (i.e., the reference or *baseline*). In this paper, two different baselines have been considered: the first baseline uses the SIBT indicated by the flight schedules as the best prediction; the second baseline goes one step further by using the most recent in-block time as derived from the ETFMS when performing the prediction.

Figure 5 shows the relative benefit of the machine learning model for the 60 random days of the 2019. The relative benefit in each cell is computed as one minus the ratio between the MAE of the machine learning model and the MAE of the baseline. The relative benefit with respect to the ETFMS and the schedules are shown in Figs. 5(a) and 5(b), respectively.

According to Fig. 5(a), the relative benefit with respect to the ETFMS ranges from 13% to 46%. For a given number of flight legs ahead, the lower the time to the SOBT of the last leg, the higher the relative benefit. These results suggest that the less uncertainty in the data, the better the model exploits the systematic patterns of the delay propagation hidden in the input features, if compared to the ETFMS. Far from the SOBT of the last leg and with many flights ahead, the uncertainty in the data is so high that, apparently, the model provides an expected value of the arrival delay based on conventional statistics, together with a large standard deviation.



(a) With respect to ETFMS



(b) With respect to Schedules

Figure 5: Relative improvement (60 random days of 2019)

When some flights of the sequence have been already flown, the data is more specific and the number of unexpected events is highly reduced. Consequently, the model is more certain about its predictions, providing an expected value closer to the actual delay and a lower standard deviation.

Figure 5(a) shows that, generally speaking, the relative improvement with respect to the schedules is higher than for the ETFMS, especially when approaching the very last flight.

2) June to December 2018: Figure 6 shows the MAE of the arrival delay for all the sequences that were operated from June to December 2018. Note that the model was trained on 285 random days of 2019, and therefore has never seen flight data of 2018. Interestingly, the performance of the model evaluated on these 7 months from the past is very similar to that shown in Fig. 4 (corresponding to 60 days of 2019), which is a good indicator of the ability of the model to learn general mechanisms of delay propagation along the sequence of flights of any aircraft, independently of the year (as long as the traffic situation is *nominal*, as discussed in next section).

Figure 7 shows the relative benefit of the machine learning model for all the sequences that were operated from June to December 2018. The relative benefit with respect to the latest ETFMS predictions is shown in Fig. 7(a). Analogously, Fig. 7(b) shows the relative benefit when taking the flight schedules as baseline, thus assuming no delays.

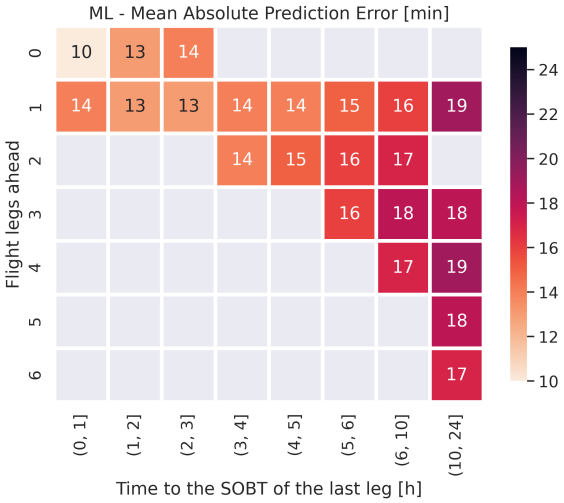


Figure 6: Arrival delay prediction MAE (June to Dec. 2018)

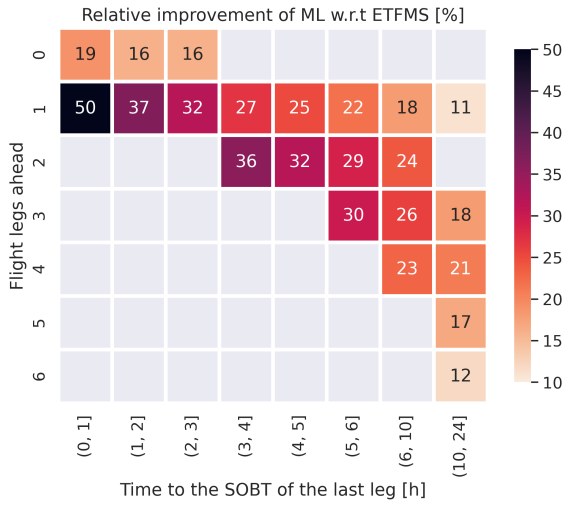
By comparing Figs. 7 and 5 it can be observed that the relative benefits of the model are very similar (independently of the baseline), despite considering very different time periods. For the time period of 2018 used in this validation exercise, the relative benefit ranges from 10% to 50% when comparing against the ETFMS, and from 11% to 70% when considering the flight schedules as hypothetical baseline.

3) January to May 2020: Assessing the performance of the model on the most recent traffic is interesting. The traffic demand during 2020, however, was exceptionally low due to the lock-down caused by the COVID-19 crisis. Despite reducing the data available for performing any kind of analysis, such atypical situation also enabled a peculiar validation exercise: to assess the performance of the model on traffic scenarios very different from those seen during training.

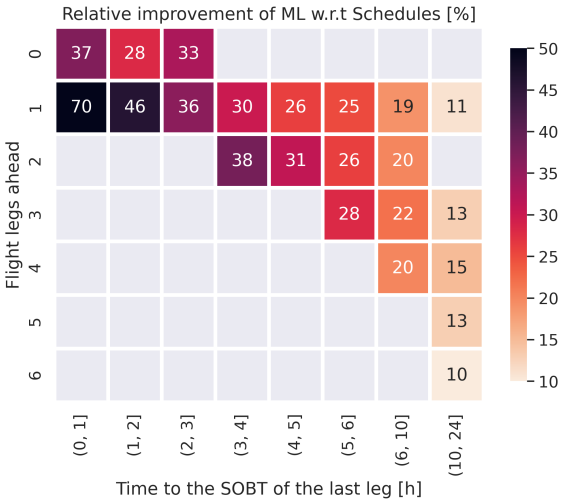
Figure 8 shows the MAE of the arrival delay predictions for all the sequences that were operated from January to May 2020. By comparing these results with those presented for 2018 and 2019, it can be observed that the performance differences between years are not significant. As mentioned before, however, the performance of the model needs to be compared with a baseline in order to quantify its added value.

The relative benefit with respect to the latest ETFMS predictions is shown in Fig. 9(a). Figure. 9(b) shows the relative benefit when taking the flight schedules as baseline.

Figure 9 shows that the relative benefits during the time period analysed for 2020 were significantly lower, if compared to 2018 and 2019. On the one hand, the performance of the ETFMS was better during 2020 due to low number of ATFM regulations, reactionary delays and unexpected events. On the other hand, the model was trained on a nominal traffic situation, where the distribution of arrival delays and their consequences on the subsequent flight legs were different.



(a) With respect to ETFMS



(b) With respect to Schedules

Figure 7: Relative improvement (June to December 2018)

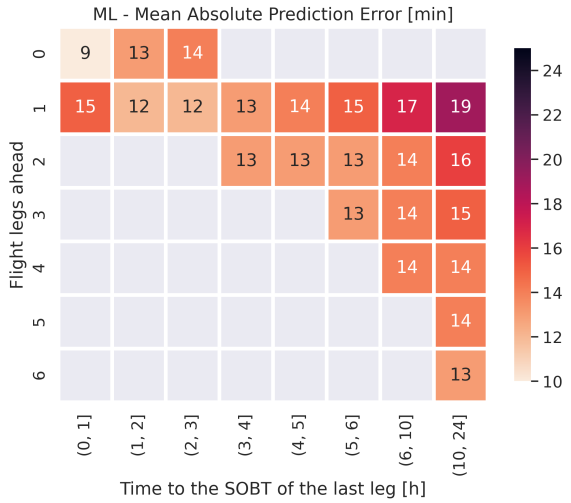
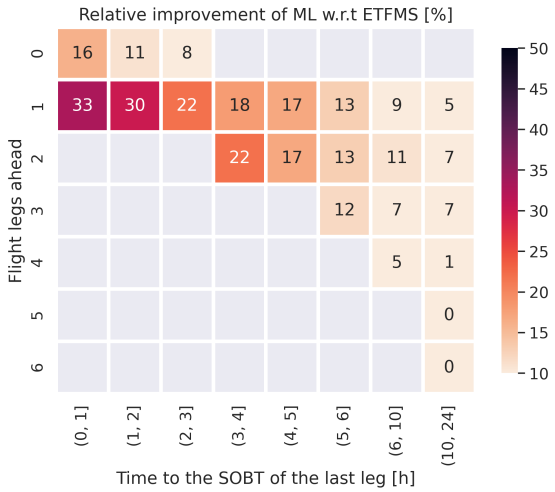
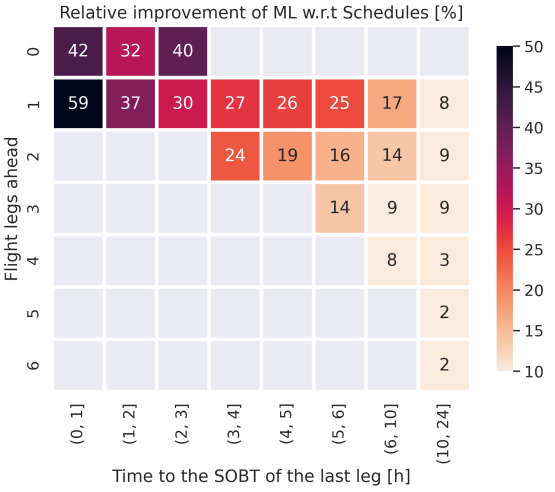


Figure 8: Arrival delay prediction MAE (January to May 2020)



(a) With respect to ETFMS



(b) With respect to Schedules

Figure 9: Relative improvement (January to May 2020)

VI. Conclusions

The model proposed in this paper uses the latest information of each flight leg of an aircraft to predict how the delays will be propagated. The distribution of delays of the last flight leg is used to assess the risk of night curfew infringement. Despite initial results from three validation exercises demonstrated notable benefits when compared to the predictions of the enhanced tactical flow management system, there is still a long way to go. The current model assumes that the flight sequences of the same aircraft at different times are independent of each-other. The architecture of the model could be enhanced by considering a two-dimensional recurrent neural network, in which the hidden state propagates along the flights and along time. Another natural extension of the model could be to explicitly consider external factors such as weather. Furthermore, in this paper all the flight sequences (independently of the last destination) were considered, aiming to train the model with a large set of examples. Future work shall compare this generic model with one tailored to those airports implementing the curfew.

Another necessary exercise in future work consists of comparing the performance of the model proposed herein with previous delay propagation models described in the literature. In fact, despite this paper is framed in the specific context of night curfew infringements, it describes a general purpose delay propagation estimator which has broad applicability beyond curfews (e.g., airline schedule integrity monitoring).

Acknowledgment

The authors would like to thank the airlines involved in this project (easyJet, Ryanair, Swiss, Transavia and Vueling) as well as Paris-Orly and Zurich airports. The authors acknowledge Laurent Renou and Franck Ballerini for their support.

References

[1] O. Eglin, M. Rotureau, P.-Y. Savidan, J.-P. Desgrange, and R. Hellot, *Different aspects of Noise Limits at Airports*, European Commission/DGTREN-F3, 10 2004, rev. 2.0.

[2] H. van Essen, B. Boon, S. Mitchell, D. Yates, D. Greenwood, and N. Porter, *Sound Noise Limits: Options for a uniform noise limiting scheme for EU airports*, CE Delft, 1 2005, rev. 1.

[3] S. AhmadBeygi, A. Cohn, Y. Guan, and P. Belobaba, "Analysis of the potential for delay propagation in passenger airline networks," *Journal of Air Transport Management*, vol. 14, no. 5, pp. 221 – 236, 2008.

[4] M. Jetzki, "The propagation of air transport delays in Europe," Master's thesis, Brüssel, 2009, aachen, Techn. Hochsch., Diplomarbeit, 2009.

[5] N. Kafle and B. Zou, "Modeling flight delay propagation: A new analytical-econometric approach," *Transportation Research Part B: Methodological*, vol. 93, pp. 520 – 542, 2016.

[6] Y. Liu and S. Ma, "Flight delay and delay propagation analysis based on bayesian network," *2008 International Symposium on Knowledge Acquisition and Modeling*, pp. 318–322, 2008.

[7] W.-B. Du, M.-Y. Zhang, Y. Zhang, and X.-B. Cao, "Delay causality network in air transport systems," *Transportation Research Part E: Logistics and Transportation Review*, vol. 118, pp. 466 – 476, 2018.

[8] N. Pyrgiotis, K. M. Malone, and A. Odoni, "Modelling delay propagation within an airport network," *Transportation Research Part C: Emerging Technologies*, vol. 27, pp. 60 – 75, 2013, selected papers from the Seventh Triennial Symposium on Transportation Analysis.

[9] L. Belcastro, F. Marozzo, D. Talia, and P. Trunfio, "Using scalable data mining for predicting flight delays," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, pp. 1 – 20, 2016.

[10] P. F. B. Campanelli *et al.*, "Modeling Reactionary Delays in the European Air Transport Network," in *4th SESAR Innovation Days*, Madrid, Spain, 2014.

[11] M. Lambelho, M. Mitici, S. Pickup, and A. Marsden, "Assessing strategic flight schedules at an airport using machine learning-based flight delay and cancellation predictions," *Journal of Air Transport Management*, vol. 82, p. 101737, 2020.

[12] D. Truong, "Using causal machine learning for predicting the risk of flight delays in air transportation," *Journal of Air Transport Management*, vol. 91, p. 101993, 2021.

[13] R. Dalmau, F. Ballerini, H. Naessens, S. Belkoura, and S. Wangnick, "Improving the Predictability of Take-off Times with Machine Learning A case study for the Maastricht upper area control centre area of responsibility," in *9th SESAR Innovation Days*, Athens, Greece, 2019.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, D. E. Rumelhart and J. L. Mcclelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.

[15] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[16] Hans Koolen and Ioana Coliban, *Flight Progress Messages Document*, EUROCONTROL, Brussels, Belgium, 2019, edition No. : 2.501.

[17] I.-K. Yeo and R. A. Johnson, "A new family of power transformations to improve normality or symmetry," *Biometrika*, vol. 87, no. 4, pp. 954–959, 2000.

[18] Y. Li and T. Yang, *Word Embedding for Understanding Natural Language: A Survey*. Cham: Springer International Publishing, 2018, pp. 83–104.