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

A Machine Learning Approach to Predict the Evolution of Air Traffic Flow Management Delay

Ramon Dalmau, Brice Genestier, Camille Anoraud, Peter Choroba & Darren Smith
Network Research Unit (NET)
EUROCONTROL Experimental Centre (EEC)
Brétigny-Sur-Orge, France

***Abstract*—In Europe, the most common air traffic flow management measure used by the network manager to resolve imbalances between demand and capacity is to impose regulations, which delay flights on ground. The ground delay assigned to a regulated flight may change from the time it is caught by the regulation(s) to the actual departure. This variability of the delay stems from the mechanisms used by the computer-assisted slot allocation system to manage the *slots* of the regulations. At present, the information on the delay evolution of a regulated flight is very limited for the airspace users, raising high uncertainty on the delay propagation and the operations management throughout the day. This paper describes the architecture of a machine learning model that, trained on historical data, is able predict the evolution of the delay for a regulated flight. Such evolution is expressed by using various indicators, which were selected by the airspace users involved in the project. The proposed model is able to predicted the trend of the delay with an accuracy of 0.75. Furthermore, results show that the model is able to reduce the prediction error (measured as the difference between the actual and the predicted delay) up to 63%, if compared to the current delay as reported by the enhanced tactical flow management system.**

***Keywords*—Machine learning; ATFM delay; CASA**

I. Introduction

According to the most likely scenario of the EUROCONTROL's statistics and forecast service, there will be around 16.2M of flights in Europe in 2040, which corresponds to 53% more traffic than in 2017 [1]. It should be noted, however, that these forecasts were made prior to the COVID-19 crisis, whose long-term consequences remain unclear. In any case, with traffic demand expected to increase in the years to come, balancing demand and capacity has become crucial.

When balancing demand and capacity, the first step is to adapt the capacity to the demand, e.g., by modifying the airspace sectorisation. If the demand cannot still be accommodated, air traffic flow management (ATFM) measures are applied to proactively align demand with capacity. In Europe, one of the habitual ATFM measures used by the network manager (NM), in coordination with the local flow management position (FMP), consists of restricting the maximum rate of aircraft entering a congested traffic volume[1] during a certain period of time, i.e, to impose a regulation.

The computer-assisted slot allocation system (CASA) creates and manages a list of ATFM slots for each regulation [2]. For example, the list of slots for a 2-hours duration regulation, with a maximum entry rate of 20 aircraft per hour, would be composed of 40 slots separated by 3 minutes.

When a regulation is activated, each flight affected by the regulation is assigned to a *pre-allocated* slot, on the basis of the first planned-first served principle. In other words, CASA sequences the regulated flights in the order they would have overflown the restricted entity in the absence of regulation.

The difference between the time of the assigned slot and the estimated time over (ETO) the restricted entity is translated into a ground delay (the ATFM delay), which added to the estimated off-block Time (EOBT)[2] determines the calculated off-block Time (COBT), i.e., the departure slot[3].

This provisional slot allocation list, however, is very likely to change over time. When CASA receives new flight data (e.g., a new flight plan or an updated ETO), it tries to assign the available slot that is closest to the ETO, i.e, the slot that produces less ATFM delay. If the slot of the ETO is free, then the flight will not receive delay. It the slot is already pre-allocated to another flight, then the slot will be given to whichever flight planned to overfly the restricted entity first. This mechanism inevitably leads to a chain reaction, since the flight whose slot has been taken attempts to obtain another slot, likely by taking the slot of another flight, and so on [2].

Only at a fixed time before the EOBT of each pre-allocated flight (typically 2 hours), the slot is finally allocated. Once the slot is allocated, it cannot be taken by another flight. However, it can improve by an automatic mechanism that continuously attempts to improve the slot and depart as close as possible to the target off-block time specified by the airline [2].

In summary, the slot of a regulated flight and the associated ATFM delay are not static, but may change over time as CASA updates the slot allocation list according to the aforementioned rules. This volatility of the ATFM delay, particularly when the slot has not yet been allocated, may cause serious predictability problems for all the stakeholders.

On the one hand, when the ATFM delay increases, the stand availability at the airport can be impacted. An aircraft whose flight is regulated might have to change its parking stand to a new one, in order to free the stand for another arriving aircraft. This implies the shift of the related ground-handling

[1] 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.

[2] The off-block time is when the aircraft pushes back the parking position.

[3] A flight may be subject to more than one regulation simultaneously. In this case, the ATFM delay of the most penalising regulation takes precedence and is forced into all other regulations to which the flight is subject.

team and equipment to the new parking stand, increasing the cost. Furthermore, the flight will not depart on time, and the delay could propagate to the next rotation of the aircraft. Such knock-on effect could even impact the entire schedule of the aircraft. Last but not least, when the information on a delay increase is not available, the airspace user cannot inform the passengers on a clear expected delay. Sometimes, when the boarding has already started, the passengers are asked to remain seated in the aircraft for an unknown duration.

On the other hand, when the ATFM delay decreases and the aircraft may depart at/close to the target off-block time, pilots and ground-handling teams must be informed about this improvement, in order to be ready on time.

This work emerged due to the need of airspace users to have a better understanding of the ATFM delay evolution when a flight is caught by one or several regulations. As discussed above, the uncertainty on the ATFM delay evolution, especially when it increases, negatively impacts the economic and operational performance of the airspace user and the airport, as well as the passenger experience.

In order to prevent these negative impacts from happening when a flight is regulated, this paper proposes several indicators that will inform the flight dispatcher and the other actors about the evolution of the ATFM delay. These indicators are provided by a machine learning model that, trained on historical data, is able to predict several hours before the COBT, the trend of the delay (i.e., the likelihood that it will increase, decrease or stay stable), its probability distribution, as well as the expected value at the actual departure time.

II. Literature review

Previous research on the field of ATFM has mainly focused on detecting and/or solving demand and capacity imbalances, using a wide variety of novel ATFM techniques, optimisation algorithms and performance metrics. For instance, Ref. [3] presented a new ATFM technique that, exploiting some of the rules and mechanisms already existing in CASA [2], could potentially allow airspace capacity to be used more effectively, thus reducing ATFM delays and their associated costs for the airspace users. A very different technique was proposed by Ref. [4], which consists of using en-route speed reduction to complement ground delay. The idea was to increase the flight time and partially perform the ATFM delay in the air, at no extra fuel cost for the airspace user.

Other works investigated optimisation algorithms to better align demand and capacity using current and/or new ATFM practices. For instance, Ref. [5] proposed a mixed-integer optimisation algorithm to assign ATFM delays in a way that minimises delay propagation to subsequent flights, simultaneously increasing flight adherence to departure slots at coordinated airports. Similarly, Ref. [6] formulated an integer programming model for strategic redistribution of flights so as to respect nominal sector capacities, in short computation times for large-scale instances. Recently, Ref. [7] proposed a collaborative ATFM framework aiming to improve the cost-efficiency for airspace users when facing ATFM regulations. One of the main difficulties of these works was to translate the ATFM delay into an monetary cost for the airspace users. Recently, [8] addressed this issue by defining quantitative and qualitative indicators to assess the expected impact of the costs that ATFM regulations could cause on airspace users.

This paper does not propose modern ATFM techniques, nor ways to optimally distribute the delays in order to minimise the cost of the airspace users. The objective of this work is to build a solution that could be deployed and provide benefits in the short-term, using the already existing ATFM practises. Taking advantage of the large amount of data collected by the NM for all flights that were regulated and all regulations that were activated, a model based on state-of-the-art machine learning techniques has been trained to perform accurate predictions of the ATFM delay evolution, therefore supporting the airspace users during their decision-making process.

In the last decade, the combination of historical data and machine learning has shown potential to improve the accuracy of predictions in many problems of air traffic management. For instance, Ref. [9] used recurrent neural networks to predict the progress of a boarding event; and Ref. [10] used gradient boosted decision trees to predict the take-off time of a flight, showing that the current ATFM delay was one of the most important input features of the model, which effectively learned that flights with a high ATFM delay often depart earlier than the current COBT, due to slot improvements.

Several works attempted to predict flight delays aggregated at a very coarse level (e.g., the whole air traffic network or particular airports), or to classify individual flights as delayed or not. For instance, Ref. [11] proposed a deep convolutional neural network architecture that, given a set of regulations and their associated maximum entry rates, is able to accurately predict the number of delayed flights and the total delay of the whole air traffic network. A supervised model capable to predict the departure delays aggregated at airport level has been also proposed by Ref. [12]. These aggregated figures could help the FMP and the NM to assess the overall consequences of certain ATFM measures, but may not help the airspace users to accurately predict the impact on the economic cost for a particular flight and the subsequent legs. Recently, Ref. [13] proposed a model based on deep learning and Levenberg-Marquart algorithm that can predict the delay as a binary indicator (i.e., yes or not) per flight.

To the best of our knowledge, this is the first work dealing with the real-time prediction of the ATFM delay for a flight that has been already caught by one or more regulations.

III. Background

The ETFMS (enhanced tactical flow management system) distributes enhanced flight data (EFD) messages to provide users with the progress and the most up-to-date state of each flight. The first EFD is sent at the moment of creating the flight plan. The EFD distribution stops when the flight terminates. An EFD message includes basic attributes of the flight (such as departure airport, destination, aircraft type, tail number, etc.), as well as the most up-to-date ATFM state (including the list of regulations to which the flight is subject, the most penalising one, the current ATFM delay, the COBT, etc.). In between the transmission of the first and the last messages, EFD are distributed whenever these

data change, and also at regular interval (currently every 60 minutes until approximately 2 hours before EOBT, and 15 minutes afterward). Further details of the ETFMS, the information included in the EFD messages, as well as the list of events that could trigger EFD are described in [14].

Therefore, for a regulated flight, the history of regulations affecting the flight (including the most penalising), the ATFM slot changes as well as the associated ATFM delay can be extracted from the ETFMS, resulting in a multi-dimensional time series composed of several categorical values (e.g., the name of the traffic volume where one of the regulations affecting the flight is applied), numerical values (e.g, the ATFM delay) as well as timestamps (e.g., the current COBT).

At a given time, a regulated flight may be subject to more than one regulation simultaneously (remember that the ATFM delay, however, is given by the most penalising regulation). In turn, a regulation may be divided in several periods, each one with a different duration and maximum entry rate. For instance, a regulation spanning from 10:00 to 12:00 could have a maximum entry rate of 30 aircraft per hour from 10:00 to 11:30, and 25 aircraft per hour from 11:30 to 12:00.

The model proposed in this paper was designed taking into account that (1) the prediction of the actual ATFM delay for a flight at a given time may be conditioned on the progress of the EFD since the flight plan creation (i.e., the *sequence* of EFD messages). For instance, if the ATFM delay is decreasing, the flight is only subject to one regulation, and the most penalising regulation has not changed for a long time, the model is likely to predict (based on the patterns learned from historical data) a different value than if the ATFM delay and the most penalising regulation have been jumping back and forth; and that (2) the ATFM information of a flight at a given time follows a *hierarchical* structure, where an EFD message could include a list of regulations, and each regulation could be decomposed in several periods.

Throughout this paper and as a general rule, scalars and vectors are denoted either with lower or upper case letters. Vectors are denoted with bold fonts, like for example $\boldsymbol{a}$, while matrices use bold computer modern calligraphy fonts, like for example $\boldsymbol{\mathcal{A}}$. Furthermore, the following notation is used when referring to sequences $(\boldsymbol{a}_t)_{t=1}^{T} = (\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_T)$.

A. Recurrent neural networks for learning sequences

Recurrent neural networks (RNN) are very popular for natural language processing (NLP) applications due to their outstanding ability to capture the underlying sequential structure present in the language, where the elements of the sequence could be characters, words or even sentences. Obviously, when aiming to predict the next word in a sentence, for instance, it is better to know which words came before it. In the same way, in order to predict the evolution of the delay, the progress of the features that determine it (e.g., the most penalising regulation) as well as their values are priceless.

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 present, LSTM (long short-term memory) and gated recurrent unit (GRU) [15] are the most popular RNN architectures because of their superior ability to deal with the vanishing gradients problem.

Let us define a generic RNN as the function 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 $(\boldsymbol{h}_t)_{t=1}^{T}$:

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

In a GRU, the following operations are performed to successively update the hidden stat at each time step t:

$$\begin{aligned}
\boldsymbol{z}_t &= \sigma\left(\boldsymbol{\mathcal{W}}_{zh}\boldsymbol{h}_{t-1} + \boldsymbol{\mathcal{W}}_{zx}\boldsymbol{x}_t\right), \\
\boldsymbol{r}_t &= \sigma\left(\boldsymbol{\mathcal{W}}_{rh}\boldsymbol{h}_{t-1} + \boldsymbol{\mathcal{W}}_{rx}\boldsymbol{x}_t\right), \\
\boldsymbol{n}_t &= \tanh\left(\boldsymbol{\mathcal{W}}_{nx}\boldsymbol{x}_t + \boldsymbol{r}_t \odot \boldsymbol{\mathcal{W}}_{nh}\boldsymbol{h}_{t-1}\right), \\
\boldsymbol{h}_t &= (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \boldsymbol{n}_t,
\end{aligned} \tag{2}$$

where the *update* gate $\boldsymbol{z}_t \in \Re^{n_h}$ controls the extent to which the information of the previous hidden state is transferred into the current hidden state, and the *reset* gate $\boldsymbol{r}_t \in \Re^{n_h}$ controls how much the previous hidden state contributes to the *new* candidate hidden state $\boldsymbol{n}_t \in \Re^{n_h}$. In Eq. (2) and for the remainder of the paper, σ represents the sigmoid function, and $\odot$ the Hadamard product. The values in the matrices[4] appearing in Eq. (2) are optimised to accomplish the task for which the recurrent neural network is trained.

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 an additional trainable parameter. Finally, one could also condition $\boldsymbol{h}_0$ on some static inputs $\boldsymbol{c} \in \Re^{n_c}$.

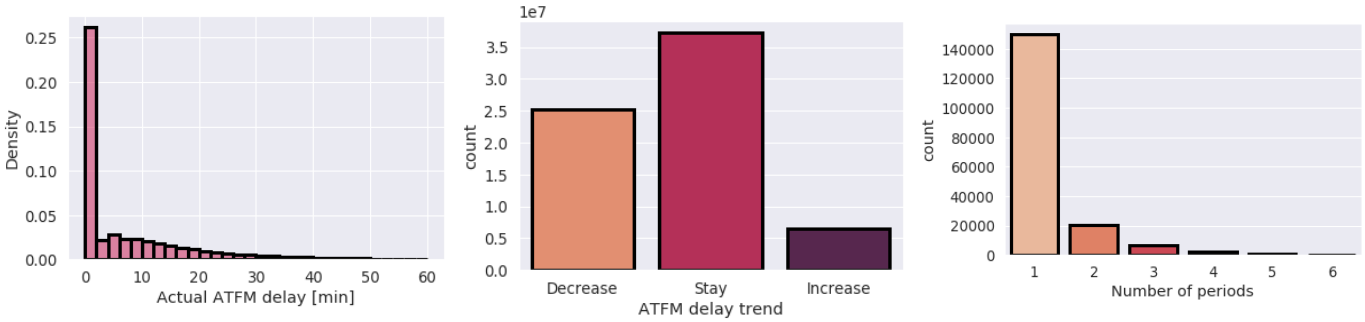
The GRU update described by Eq. (2) allows for control of the flow of information along the sequence, from the past to the future. This implies that $\boldsymbol{h}_t$ is an implicit function of $(\boldsymbol{x}_q)_{q=1}^{q<t}$. For many tasks, however, it is beneficial to have access to past as well as future information. 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 order. 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:

$$\boldsymbol{h}_t = \overrightarrow{\boldsymbol{h}_t} \parallel \overleftarrow{\boldsymbol{h}_t}. \tag{3}$$

IV. Mathematical Model

Whenever the user aims to predict the ATFM evolution for a regulated flight, the following steps are performed: (1) collect all previous EFD messages received for that flight; (2) For each EFD message, extract the set of features listed in Section V, in order to generates the inputs for the model; (3) call the model described below to generate a sequence of hidden states $(\boldsymbol{h}_t)_{t=1}^{T}$, each one associated with one of the previous messages; (4) keep only the latest hidden state of the sequence $\boldsymbol{h}_T$ (i.e., that corresponding to the most recent message) and then predict the desired indicator(s).

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



(a) Distribution of actual ATFM delay

(b) Distribution of ATFM delay trend

(c) Number of periods per regulation

Figure 1: Statistics of regulated flights and ATFM regulations during the 2019 time period

The architecture of the generic model presented herein is inspired by the hierarchical attention neural network proposed by Ref. [16]. The fundamental idea of this model is that the inputs follow a hierarchical structure: a regulation is divided in periods, a message may include several regulations, and the sequence of messages is needed to perform the prediction. Most of the hierarchical models proposed in the literature combine RNNs and attention mechanisms. For the task addressed in this paper, however, initial assessments revealed that attention mechanisms did not boost the performance of the model, but significantly increased the complexity of the architecture. A complex architecture is more difficult to train, debug and maintain. For this reason, the model presented in this section uses only bi-directional RNNs to capture the most important information of each level in the hierarchy and update the next one, starting from the periods of the regulations (first level) to end up with the final prediction.

Let us define $\boldsymbol{x}_{trp}$ as the features for one of the P_{tr} periods that compose the regulation r, which was affecting the flight when the EFD message t was received. Following the notation adopted in this paper, $(\boldsymbol{x}_{trp})_{p=1}^{P_{tr}}$ represents the sequence of features for the P_{tr} periods, ordered by ascending starting time of the period. Note that P_{tr} could change with r and t.

Analogously, let us define $\boldsymbol{x}_{tr}$ as the features for one of the R_t regulations affecting the flight when the EFD message t was received; and $(\boldsymbol{x}_{tr})_{r=1}^{R_t}$ as the sequence of features for the for the R_t regulations, ordered by ascending ETO the restricted entity. Note that R_t could change with t, and that only one of these regulations is the most penalising. Finally, let us define $\boldsymbol{x}_t$ as the features of the EFD message t.

When processing the first level of the hierarchy, each $\boldsymbol{x}_{tr}$ shall updated based on the information captured from the corresponding sequence of periods $(\boldsymbol{x}_{trp})_{p=1}^{P_{tr}}$. Typically, however, P_{tr} is 1 or 2 (see Fig. 1c). For this reason, the processing of the first hierarchy could be simplified to:

$$\tilde{\boldsymbol{x}}_{tr} = \boldsymbol{x}_{tr} \parallel \boldsymbol{x}_{trq}, \tag{4}$$

where q is the time period of the regulation r in which the ETO the restricted entity is contained.

Then, the feature vector of each EFD message $\boldsymbol{x}_t$ is updated based on the corresponding sequence of regulations $(\tilde{\boldsymbol{x}}_{tr})_{r=1}^{R_t}$. First, the sequence of regulations is passed thorugh a BiGRU in order to obtain a representation of the sequence:

$$\left(\tilde{\boldsymbol{h}}_{tr}\right)_{r=1}^{R_t} = \text{BiGRU}\left((\tilde{\boldsymbol{x}}_{tr})_{r=1}^{R_t}, \overrightarrow{\tilde{\boldsymbol{h}}_{t0}}, \overleftarrow{\tilde{\boldsymbol{h}}_{t0}}\right), \tag{5}$$

where the initial hidden states of the forward and backward GRUs are conditioned on the content of the EFD message:

$$\begin{aligned} \overrightarrow{\tilde{\boldsymbol{h}}_{t0}} &= \text{FFNN}\left(\boldsymbol{x}_t\right) \\ \overleftarrow{\tilde{\boldsymbol{h}}_{t0}} &= \text{FFNN}\left(\boldsymbol{x}_t\right). \end{aligned} \tag{6}$$

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 $\boldsymbol{x} \in \Re^{|\boldsymbol{x}|}$:

$$\boldsymbol{y} = \phi\left(\boldsymbol{\mathcal{W}}\boldsymbol{x} + \boldsymbol{b}\right), \tag{7}$$

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 $\overrightarrow{\tilde{\boldsymbol{h}}_{t0}}$ are different from those of the FFNN that generates $\overleftarrow{\tilde{\boldsymbol{h}}_{t0}}$. The latter statement applies all along the paper, i.e., each FFNN has its own (unknown) set of parameters, which will be fitted to the historical data during the training process. 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 VI-B.

The message is concatenated with the last hidden state of the BiGRU in order to generate the updated message:

$$\tilde{\boldsymbol{x}}_t = \text{FFNN}\left(\boldsymbol{x}_t \parallel \tilde{\boldsymbol{h}}_{tR_t}\right). \tag{8}$$

The messages updated with information of the corresponding regulations are then processed by a conventional GRU:

$$(\boldsymbol{h}_t)_{t=1}^{T} = \text{GRU}\left((\tilde{\boldsymbol{x}}_t)_{t=1}^{T}, \boldsymbol{h}_0\right), \tag{9}$$

where the initial hidden state $\boldsymbol{h}_0$ is an additional trainable parameter. The hidden state corresponding to the most recent message at the moment of performing the prediction (i.e., $\boldsymbol{h}_T$) is used to predict the evolution of the ATFM delay.

In this paper, the evolution of the ATFM delay is expressed by using three indicators: the expected value of the actual ATFM delay (in minutes), its probability distribution, and the trend. Each one of these indicators is predicted by a *copy* of the neural network architecture described above. The only difference between these three copies is in the last layers, which process $\boldsymbol{h}_T$ to predict the specific indicator. Note that since all the three models aim to predict a very similar value, sharing the parameters of the initial layers may benefit and accelerate the training process. In the experiment performed herein, however, the trainable parameters of each model have been optimised to minimise the loss function from scratch. Next sections describe the final layers for each indicator, as well as the loss function used to optimise the parameters.

A. Basic regression to predict the expected value

The most evident approach to provide an indicator of the ATFM delay evolution consists of predicting the expected value of the actual ATFM delay (in minutes). In this case, the output $y_{\text{basic}} \in [0, \infty)$ of the model would be given by:

$$y_{\text{basic}} = \text{FFNN}\left(\boldsymbol{h}_T\right), \tag{10}$$

where the $\text{ReLU}\left(x\right) = \max\left(0, x\right)$ activation function is used in the very last layer to ensure that $y_{\text{basic}} \in [0, \infty)$, i.e., that the expected value of the actual ATFM delay is positive.

The training of the model can be addressed as a regression problem, in which the parameters of the whole neural network are optimised to minimise the mean absolute error (MAE) between the predicted and the actual ATFM delay:

$$L = |y_{\text{basic}} - y_{\text{true}}|. \tag{11}$$

After being trained with historical data, the basic regression model could predict the actual ATFM delay, yet the user has no means to quantify the uncertainty of the indicator provided by the model. This limitation inspired the development of a model that predicts the probability distribution of the actual ATFM delay (rather than just its expected value), aiming to better inform the user about the likelihood of the prediction.

B. Poisson regression to predict the probability distribution

The objective of this model is to predict the probability distribution that best describes the actual ATFM delay.

The number of occurrences that the actual ATFM delay is zero, however, is so large that its empirical distribution do not readily fit conventional functions (e.g., Gaussian, Poisson, Binomial). This situation is known in the literature as zero-inflated process [17]. Figure 1a illustrates this behaviour.

Let us imagine that, for each regulated flight, there are two possible cases: (1) With probability ρ, the current ATFM delay will decrease to zero; and (2) With probability $1-\rho$, the actual ATFM delay is generated from a Poisson distribution with rate $\lambda \in [0, \infty)$. Then, the probability distribution of a zero-inflated Poisson (ZIP) random variable y is:

$$\text{ZIP}(y \mid \lambda, \rho) = \begin{cases} \rho + (1-\rho)e^{-\lambda} & \text{if } y = 0 \\ (1-\rho)\text{Pois}\left(y, \lambda\right) & \text{otherwise} \end{cases} \tag{12}$$

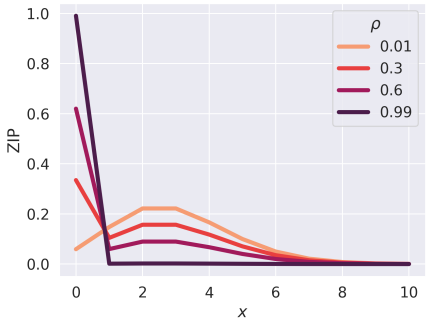
where the probability distribution of the standard Poisson distribution is:

$$\text{Pois}\left(y \mid \lambda\right) = \frac{e^{-\lambda}\lambda^y}{y!} \tag{13}$$

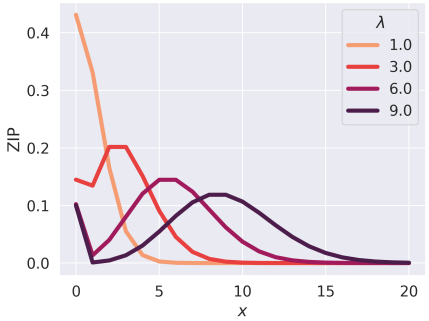
The expected value of the ZIP distribution is given by:

$$\mathbb{E}\left[\text{ZIP}(\cdot \mid \lambda, \rho)\right] = (1-\rho)\lambda \tag{14}$$

Figure 2a shows how the ZIP distribution changes with ρ, for a fixed λ. Analogously, Figure 2b shows how the ZIP distribution changes with λ, for a fixed ρ.



(a) $\lambda = 3$



(b) $\rho = 0.1$

Figure 2: ZIP probability distribution

Note that the ZIP distribution is defined by two parameters: λ and ρ. Since the objective of the model is to predict the parameters of the ZIP distribution that maximises the likelihood of the actual ATFM delay, the neural network has two outputs, which are generated by different FFNNs:

$$\begin{aligned} \lambda &= \text{FFNN}\left(\boldsymbol{h}_T\right) + \epsilon \\ \rho &= \text{FFNN}\left(\boldsymbol{h}_T\right) \end{aligned} \tag{15}$$

where the ReLU activation function is used in the last layer of the FFNN that generates λ to ensure that $\lambda \in [0, \infty)$; and the sigmoid (σ) activation function in the last layer of the FFNN that generates ρ to ensure that $\rho \in [0, 1]$. Note that a small constant value ϵ is added to the prediction of the model in order to comply with the support of the parameter λ.

Finally, the parameters of the resulting model are optimised to minimise the negative log-likelihood:

$$L = -\log \text{ZIP}\left(y_{\text{true}} \mid \lambda, \rho\right) \tag{16}$$

C. Ordinal regression to classify the trend

The last model emerged from the request of the airspace users to have an indicator of the ATFM delay trend of a given flight. That is, the probability that the current ATFM delay of the flight will increase, decrease or stay stable[5]. This question could be addressed as a classification problem, where the model generates as many outputs as classes (Decrease, Stay and Increase), and each output represents the probability of the true delay trend belonging to the associated class.

Different from traditional multi-class classification problems (e.g., determining if a picture contains a dog, a cat or a horse), the targets proposed in this paper have ordinal relationships to each-other. That is, there is a larger difference between Decrease and Increase than between Decrease and Stay. Similarly, from the point of view of the model, wrong predictions should also have ordinal relationships. In other words, the penalty given to the model when it predicts Increase and the true value is Decrease must be higher than when the prediction is Stay. This class of problems is commonly known in the literature as ordinal regression [18].

For K classes, the basic idea behind ordinal regression is to learn how to *split* the prediction space according to a sequence of $K-1$ cutpoints $(c_k)_{k=1}^{K-1} = (c_0, c_1, \dots c_{K-1})$. The probability that an observation y belongs to the class k is given by the cumulative logistic link (CLL) function:

$$\text{CLL}(y = k \mid (c_k)_{k=1}^{K-1}) = \begin{cases} \sigma\left(c_0 - y\right) & \text{if } k = 0 \\ \sigma\left(c_k - y\right) - \sigma\left(c_{k-1} - y\right) & \text{if } 0 < k < K \\ 1 - \sigma\left(c_{K-1} - y\right) & \text{if } k = K \end{cases} \tag{17}$$

The cutpoints become additional parameters to be learned during the training process using back-propagation. Varying the cutpoints changes the class probabilities for a given input. Figure 3 shows the CLL with three classes (Decrease, Stay and Increase) for two different locations of the cutpoints.

The input of the CLL $y_{\text{ordinal}} \in (-\infty, \infty)$ is given by a FFNN that processes $\boldsymbol{h}_T$, using a linear activation function:

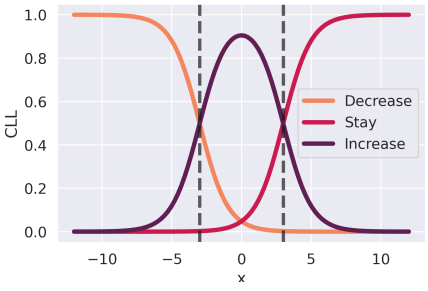
$$y_{\text{ordinal}} = \text{FFNN}\left(\boldsymbol{h}_T\right) \tag{18}$$

Similar to the Poisson regression model, the loss function is the negative log-likelihood:

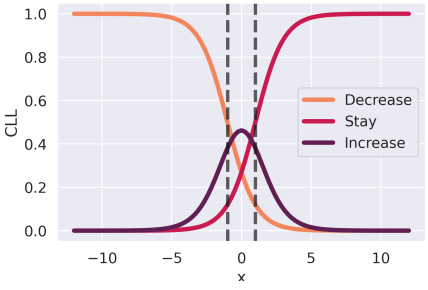
$$L = -\log \text{CLL}\left(y_{\text{ordinal}} = y_{\text{true}} \mid (c_k)_{k=1}^{K-1}\right) \tag{19}$$

The frequency of these three classes, however, is not balanced (see Fig. 1b). This characteristic often results in models that have poor predictive performance for the minority class. If no measure is taken to address the imbalance problem (e.g., down-sampling, weighting the examples in the loss function and/or during sampling), the trained model may have difficulties to identify situations in which the delay increases.

[5]In this paper, the ATFM delay is considered stable if it does not change more than 5 minutes. Note that this threshold was selected by the airspace users involved in the project, based on their operating methods.



(a) $(c_k)_{k=1}^{2} = (-3, 3)$



(b) $(c_k)_{k=1}^{2} = (-1, 1)$

Figure 3: CLL probability distribution (dashed lines indicate the position of the cutpoints

V. Features

Table I lists the input features used by the model to make predictions, classified according to the type of transformation that is applied and the level in the hierarchy of features.

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 models, like decision trees, are robust to arbitrary scaling of the numerical features and only require categorical features to be encoded as integers (e.g., VLG $\rightarrow$ 1). Neural networks, on the other hand, prefer numerical features to be standardised or normalised, and categorical features take special precautions.

Normalisation consists of re-scaling a feature from the original range to a smaller range, typically through dividing by the maximum absolute value, or by performing a *min-max* transformation. 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. In this paper, the well-known Yeo-Johnson transformation [19] has been applied to some features, followed by a standarisation.

Other features related with time (e.g., hour of the day) are cyclical in nature. A common method for encoding cyclical features is to map the data into two dimensions, using the sine and cosine transformations. The boolean indicators simply indicate whether a given condition holds or not.

TABLE I. Features of the model

Feature Description	Transformation	Level in the hierarchy
Aircraft type (e.g., A320) Departure airport (e.g., LFPO) Destination airport (e.g., LEBL) Type of event that triggered the EFD message [14] (e.g., CASA slot allocation / update) Class of event that triggered the EFD message [14] (e.g., EFD generated by a slot re-calculation event) ATFM state of the flight [14] (e.g., The flight is regulated, but the slot has not yet been published) Type of flight [14] (e.g., scheduled; non-scheduled, military, etc.) Ready state [14] (e.g., ready for slot improvement, ready to depart, etc.) Aircraft operator (e.g., VLG) Airline (e.g., VLG) Type of departure airport in CDM, advanced air traffic control (ATC) tower, or conventional Collaborative decision making (CDM) state [14] (e.g., the flight is pre-sequenced by ATC)	Embedding	Message ($\boldsymbol{x}_t$)
Traffic volume where the regulation is applied [2] Geographical entity associated to the traffic volume [2] Type of geographical entity [2] (e.g., airspace, aerodrome, etc.) Reason of the regulation [2] (e.g., ATC capacity, weather, etc.)		Regulation ($\boldsymbol{x}_{tr}$)
Estimated taxi-in time at departure airport Current ATFM delay Time to EOBT Time from EOBT of the flight plan to current EOBT Time from previous EOBT to current EOBT Time since previous EFD message Time since creation of the flight plan Time from EOBT to TOBT Time from EOBT to TSAT Duration of the flight	Power → Standarisation	Message ($\boldsymbol{x}_t$)
Entry time difference with respect to entry time at previous regulation Time to start time of the regulation Time since activation time of the regulation Time since last change time of the regulation Duration of the regulation Time since the first time that the flight is subject to the regulation Time since the previous time that the flight is subject to the regulation Time since the regulation is (or is not) the most penalising Window width of the regulation [2]		Regulation ($\boldsymbol{x}_{tr}$)
Pending entry rate of the regulation [2] Entry rate of the regulation [2]		Period ($\boldsymbol{x}_{trp}$)
Hour of the EOBT of the flight plan Day of week of the EOBT of the flight plan Month of the EOBT of the flight plan	Cyclic	Message ($\boldsymbol{x}_t$)
Number of regulations affecting the flight	Normalisation	Message ($\boldsymbol{x}_t$)
Time from start time of the regulation to entry time[1] Time from start time of the regulation to exit time[1] Maximum flight level of the regulation Minimum flight level of the regulation Number of periods that compose the regulation		Regulation ($\boldsymbol{x}_{tr}$)
Time from start time of the regulation to end of the period[1] Time from end of the period to end time of the regulation[1]		Period ($\boldsymbol{x}_{trp}$)
Is the TSAT present?	Boolean missing indicator	Message ($\boldsymbol{x}_t$)
Is it the most penalising? Is it the first regulation crossed by the flight? Does it allow ATFM slot improvement? Does it allow ATFM slot deterioration?		Regulation ($\boldsymbol{x}_{tr}$)

Finally, each categorical feature has been encoded by using an independent embedding layer [20]. Each embedding layer automatically learns the representation of each category of the corresponding feature in a multi-dimensional space. The result of the embedding layer is a vector of continuous values for each category, which values are closer for categories with similar effect for the task the model is trained.

[1]This feature is normalised with respect to the duration of the regulation

VI. Results

The model presented in Section IV, which uses the features listed in Section V to predict several indicators of the ATFM delay evolution, has been trained on real flight data. Section VI-A describes the setup of the experiment. Section VI-B lists the final hyper-parameters of the model. An illustrative examples is shown in Section VI-C. Finally, the aggregated performance metrics are discussed in Section VI-D.

A. Setup of the experiment

The data used in this study includes all flights that were regulated during 2019. The training examples have been obtained from 300 random days of 2019 (with around 2.2M regulated flights). From the remaining days of 2019, 20 days (with around 120K regulated flights) were used for early stopping and hyper-parameters tuning (i.e., the validation set), and the other 40 days (with around 300K regulated flights) to assess the performance of the model on unseen data. Three days out of the 40 in the test set were selected by the airspace users, which were interested on quantifying the performance of the model in scenarios with a large number of regulated aircraft due to, e.g., air traffic control (ATC) strikes or severe issues related to weather. Table II shows the key metrics of the three days selected for the validation exercise.

TABLE II. Key metrics of the three days selected for the validation exercise with the airspace users

Date	#Flights	ATFM delay	Represents
30th of April 2019	30890	47260 min	Normal scenario
6th of December 2019	28375	141150 min	ATC industrial action
27th of July 2019	32524	320215 min	Summer thunderstorms

B. Hyper-parameters of the model

Table III lists the hyper-parameter of the model, which were selected based on expert judgment and manual fine-tuning using the validation set. 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 mentioned, the activation function of each layer is the ReLU.

TABLE III. Hyper-parameters of the model

Hyper-parameter	Value
Batch size / Learning rate / Epochs	32 / 1e-4 / 4
Gradient clipping value	5
Min. / Max. embedding size	4 / 64
Hidden state size of GRU in Eq. (9)	256
Number of recurrent layers in Eq. (9)	2
Hidden state size of BiGRU in Eq. (5)	128
Number of recurrent layers in Eq. (5)	1
FFNNs in Eqs. (6) for hidden states	128 (ReLU)
FFNN in Eqs. (10) and (18) for y	256$\rightarrow$128$\rightarrow$64$\rightarrow$32$\rightarrow$1 (linear)
FFNN in Eq. (15) for λ	256$\rightarrow$128$\rightarrow$64$\rightarrow$32$\rightarrow$1 (ReLU)
FFNN in Eq. (15) for ρ	256$\rightarrow$128$\rightarrow$64$\rightarrow$32$\rightarrow$1 (sigmoid)

The training of each one of the models (with around 1.8M of parameters each) took around 7 h using a NVIDIA Tesla K80 GPU and the Intel Xeon E5-2690 v3 (Haswell) processor.

C. Illustrative example

Table IV shows an illustrative example of the evolution of the ATFM delay and the indicators of the model for a regulated flight extracted from the test set. The left region of the table show the information that, nowadays, can be obtained from the ETFM: the time to EOBT, the state of the flight[6] and the current ATFM delay. The right region of the table shows the indicators provided by the trained models: the expected ATFM delay (y_{basic}), the two parameters of the ZIP distribution (λ and ρ) and the probability that the ATFM delay will decrease, stay stable or increase (p_d, p_s and p_i, respectively). Each row corresponds to an EFD message.

TABLE IV. Illustrative example

Time to EOBT	State	Delay	y_{basic}	λ	ρ	p_d	p_s	p_i
03:43:34	FS	79	12	23	0.25	0.98	0.02	0
03:31:37	FS	64	10	23	0.21	0.97	0.03	0
03:30:54	FS	79	11	26	0.19	0.98	0.02	0
02:58:47	FS	76	11	26	0.17	0.98	0.02	0
02:39:55	FS	46	09	21	0.20	0.93	0.07	0
02:28:42	FS	76	11	25	0.17	0.98	0.02	0
02:07:22	FS	76	12	26	0.16	0.98	0.02	0
02:02:54	SI	76	12	26	0.16	0.98	0.02	0
02:00:47	SI	76	12	25	0.17	0.98	0.02	0
00:42:48	SI	76	21	30	0.11	0.97	0.03	0
00:39:45	SI	76	21	31	0.12	0.96	0.04	0
00:39:38	SI	76	23	31	0.12	0.96	0.04	0
00:35:07	SI	76	31	39	0.08	0.96	0.04	0
00:23:46	SI	34	22	23	0.05	0.80	0.20	0
00:15:46	SI	19	15	14	0.08	0.61	0.37	0
00:09:27	SI	64	34	40	0.03	0.93	0.07	0
00:07:31	SI	34	23	24	0.03	0.74	0.26	0
00:01:52	SI	19	15	15	0.06	0.54	0.43	0
00:01:51	SI	19	16	16	0.03	0.41	0.55	0

The flight shown in Table IV was caught by a regulation 3 h and 43 min before EOBT, and was assigned with a delay of 79 min by CASA. At the same time, the model was predicting an actual ATFM delay of 12 minutes, with a high probability (98%) that the delay would decrease by more than 5 min, and a probability that it would decrease to 0 (i.e., ρ) of 25%.

While the slot of the flight was still pre-allocated (FS state), the ATFM delay fluctuated between 46 and 79 minutes. This variability stems from the slot changes caused by the rules that govern the CASA algorithm. Despite the ATFM delay variations, the indicators provided by the model were very stable. The prediction for the actual ATFM delay remained in the range 9 to 12 min, the model was still very confident that the delay would decrease, yet the probability that it would decrease to 0 dropped to 16%. Then, the slot was finally allocated to the flight (i.e., it reached the SI state).

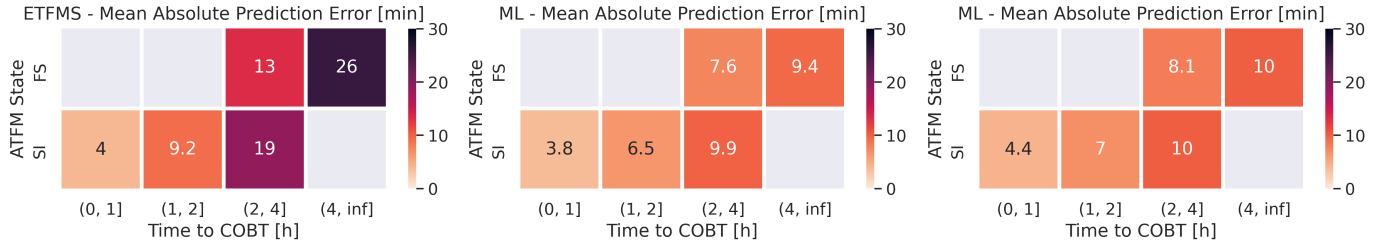
42 min before EOBT, the ATFM delay stabilised at 76 min. For this reason, the model became less optimistic and slightly increased the actual ATFM delay prediction to 21 min. Note, however, that the model was convinced that, before departure, the ATFM delay would decrease more than 5 min.

This prediction became a reality 23 min before EOBT, when the ETFM delay decreased to 34 min. Actually, this flight had a final ATFM delay of 19 min, closer to the 12 min predicted by the model almost 4 h before, when the ATFM delay was 79 min. These indicators provide valuable information to assess the evolution of the ATFM delay.

D. Aggregated performance metrics

This section shows the aggregated performance metrics evaluated in the 40 days of the test set. Results for the regression models (basic and Poisson) are presented in Section VI-D1; while Section VI-D2 shows the performance metrics for the trend classification (ordinal regression) model.

[6]FS stands "for Filed Slot allocated": The flight is regulated, but the ATFM slot has not yet been published; and SI stands for "Slot Issued": The flight is regulated and the ATFM slot has been published.



(a) Current delay reported by the ETFMS (b) Basic regression model (c) Poisson regression model

Figure 4: Mean Absolute Error (evaluated on the test set)

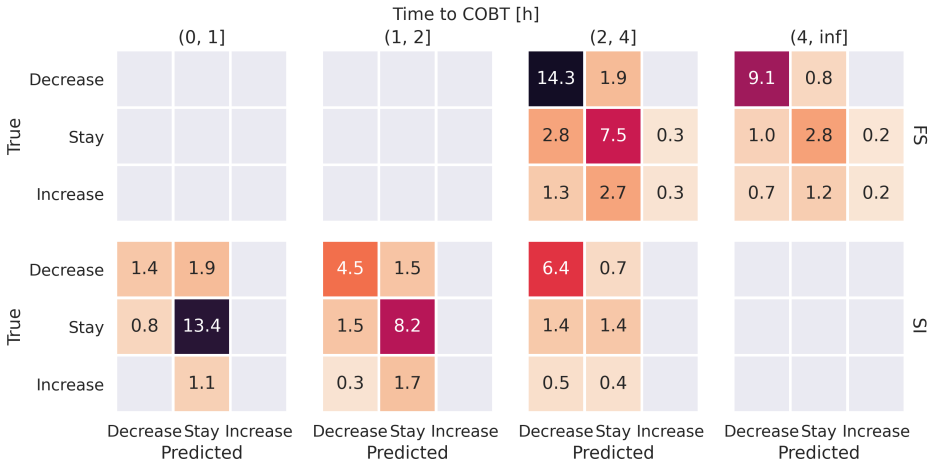


Figure 5: Confusion matrix (evaluated on the test set)

1) Basic and Poisson regression: Figure 4 shows the MAE between the actual ATFM delay and: the current delay as reported by the ETFMS (Fig. 4a), the actual delay predicted by the basic regression model (Fig. 4b), or the expected value of the actual delay distribution predicted by the Poisson regression model (Fig. 4c). Remember that, for the latter case, $\mathbb{E}\left[\text{ZIP}(\cdot \mid \lambda, \rho)\right] = (1-\rho)\lambda$. The MAE letric is aggregated for all messages of all flights in the test set. Results are shown as a function of the time to COBT (horizontal axis) and the ATFM state of the flight (vertical axis). Note that the meaning of FS and SI states were presented in the previous section.

According to Fig. 4a, the MAE of the ETFMS ranges from 4 to 26 min, depending on the look-ahead time of the prediction and the ATFM state of the flight. For instance, more than 4 hours before the COBT, when the flight is regulated but the slot has not yet been allocated (FS state), the ATFM delay reported by the ETFMS is far from the one that will actually happens. Between 2 and 4 hours, the prediction error of the ETFMS is reduced by almost 50%. Once the slot is allocated (typically 2 hours before EOBT), the slot does not change so often and, consequently, the delay reported by the ETFMS is already accurate (the error is lower than 10 min).

More than 4 hours before the COBT, both regression models are able to reduce the MAE of the predictions to around 10 minutes (63% improvement if compared to the ETFMS). These predictions do not improve significantly when moving to 2-4 hours before COBT, yet the improvement with respect to the ETFMS in the same conditions is still notable (45%). The slot of a regulated flight is typically allocated two hours before EOBT. Then, CASA tries to improve the slot through the true revision process. Figure 4 shows that the relative benefit of the machine learning approach after slot allocation is under 30%. In fact, one hour or less before COBT, the delay reported by the ETFMS is similar to that predicted by the two regression models. Note, however, that the Poisson regression model was not explicitly trained to minimise the MAE, thus the comparison with other models using this metric is not fair. The Poisson model could be less accurate when predicting the expected value, but it provides the probability distribution of actual ATFM delay, which cannot be extracted from the predictions of the basic regression model.

2) Trend classification: Figure 5 shows the confusion matrices for the predictions of the trend classification model on the test set, aggregated by time to COBT and state of the flight. The rows and the columns of each confusion matrix correspond to the true and the predicted class, respectively. Diagonal and off-diagonal cells correspond to correctly and incorrectly classified samples, respectively. The number in each cell indicates the frequency of samples with respect to the messages of the 300K regulated flights in the test set.

For instance, 17.8% of the samples belong to predictions performed between 1 and 2 hours before COBT, when the slot was allocated (SI state). Under these circumstances, in 34% of the corresponding samples the delay actually decreased. From this 34%, the model was correct 79% of the cases, and it predicted a stable delay in the remaining 21%.

Results suggest that the model is very accurate when predicting whether the delay is going to decrease or stay stable, even far away from COBT. As discussed in Section IV-C, however, the imbalance present in the target hampers the correct identification of the minority class: delay increase.

The model has an overall recall of 0.75, with a precision of 0.73 and a F1-score of 0.71. Interestingly, the precision of an hypothetical model that always predicts *stay stable* would have a recall of 0.46, a precision of 0.21 and a F1-score of 0.29. It should be noted that these metrics do not fully represent the effectiveness of the model in an operational environment. The standard metrics (i.e., recall, precision, F1-score) assume that a prediction is either correct or wrong. As discussed in Section IV-C, however, the model was trained to perform ordinal regression, i.e., the penalty for a wrong prediction depends on how *far* it is from the true value. This fact can be observed in Figure 5: when the model performs a wrong prediction it typically assigns the closest class.

VII. Conclusions

This paper proposed a generic machine learning model inspired by hierarchical architectures that, trained on historical data, is able to provide several indicators of the air traffic flow management (ATFM) delay evolution of each individual flight. The model can be used as soon as the flight is caught by any regulation, providing up-to-date indicators to the airspace users all along the progress of the flight. These indicators were designed to improve their situation-awareness, and thus be able to mitigate the negative impacts of the ATFM delay.

Two variants of model, designed to predict the actual ATFM delay and its probability distribution, respectively, demonstrated to improve the accuracy of the current delay predictions from 30% to 63%, more than 2 h before calculated off-block time (COBT). When approaching COBT, however, the ATFM delay does not present significant variability, and the relative benefits of the proposed models are residual.

Another variant of the model explicitly designed to predict the trend of the ATFM delay showed an overall accuracy of 0.75. This variant effectively captures when the ATFM delay is going to decrease or stay stable, yet facing some difficulties to predict when the delay is going to increase.

In future work, the model proposed herein should be compared with other models existing in the scientific literature (if any) to predict the ATFM delay and its trend for individual flights, and not only with the current ETFMS predictions.

The proposed model performs the predictions per flight, based only on the sequence of the features extracted from its corresponding flight progress messages. In other words, the model does not explicitly consider the effects that the overall air traffic network have on its ATFM delay evolution. Future work should explicitly consider these information as new features of the model, e.g., by including the evolution of the entry counts at the traffic volumes associated to the regulations to which the flight is subject, or by using a more complex graph architecture that makes predictions for the whole set of regulated flights in the network simultaneously.

Last but not least, strategies to improve the performance of the trend classification model and detect more cases with delay increase could be investigated, e.g., by giving more weight to examples belonging to this class in the loss function or by sampling them with higher probability during training.

Acknowledgement

The authors would like to thank the airlines involved in this project (Air France, easyJet, Ryanair, Swiss, Transavia, Vueling and Wizzair). The authors also acknowledge Laurent Renou and Franck Ballerini for their inconditional support.

References

[1] *European Aviation in 2040: Challenges of growth*, EUROCONTROL, October 2018.

[2] *ATFCM Users Manual*, EUROCONTROL, November 2018, rev. 22.1.

[3] S. Ruiz, H. Kadour, and P. Choroba, "An innovative safety-neutral slot overloading technique to improve airspace capacity utilisation," in *SESAR Innovation Days (SID) 2019*, Athens, Greece, December 2019.

[4] L. Delgado and X. Prats, "En route speed reduction concept for absorbing air traffic flow management delays," *Journal of Aircraft*, vol. 49, pp. 214–224, 01 2012.

[5] N. Ivanov, F. Netjasov, R. Jovanović, S. Starita, and A. Strauss, "Air traffic flow management slot allocation to minimize propagated delay and improve airport slot adherence," *Transportation Research Part A: Policy and Practice*, vol. 95, pp. 183–197, 2017.

[6] T. Bolić, L. Castelli, L. Corolli, and D. Rigonat, "Reducing atfm delays through strategic flight planning," *Transportation Research Part E: Logistics and Transportation Review*, vol. 98, pp. 42–59, 2017.

[7] Y. Xu, R. Dalmau, M. Melgosa, A. Montlaur, and X. Prats, "A framework for collaborative air traffic flow management minimizing costs for airspace users: Enabling trajectory options and flexible pre-tactical delay management," *Transportation Research Part B: Methodological*, vol. 134, pp. 229 – 255, 2020.

[8] L. Delgado, G. Gurtner, T. Bolić, and L. Castelli, "Estimating economic severity of air traffic flow management regulations," *Transportation Research Part C: Emerging Technologies*, vol. 125, p. 103054, 2021.

[9] M. Schultz and S. Reitmann, "Machine learning approach to predict aircraft boarding," *Transportation Research Part C: Emerging Technologies*, vol. 98, pp. 391–408, 2019.

[10] R. Dalmau, F. Ballerini, H. Naessens, S. Belkoura, and S. Wangnick, "An Explainable Machine Learning Approach to Improve Take-off Time Predictions," *Journal of Air Transport Management*, vol. 0, pp. 0–0, 2021.

[11] R. Sanaei, B. A. Pinto, and V. Gollnick, "Toward atm resiliency: A deep cnn to predict number of delayed flights and atfm delay," *Aerospace*, vol. 8, no. 2, 2021.

[12] B. Ye, B. Liu, Y. Tian, and L. Wan, "A methodology for predicting aggregate flight departure delays in airports based on supervised learning," *Sustainability*, vol. 12, no. 7, 2020.

[13] M. F. Yazdi, S. R. Kamel, S. J. S. M. Chabok, and M. Kheirabadi, "Flight delay prediction based on deep learning and levenberg-marquart algorithm," *Journal of Big Data*, vol. 7, no. 1, p. 106, 2020.

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

[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] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 1480–1489.

[17] W. Tu, *Zero-Inflated Data*. American Cancer Society, 2006.

[18] F. Pedregosa, F. Bach, and A. Gramfort, "On the Consistency of Ordinal Regression Methods," *Journal of Machine Learning Research*, vol. 18, pp. 1–35, 2017.

[19] 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.

[20] A. Mandelbaum and A. Shalev, "Word Embeddings and Their Use In Sentence Classification Tasks," *arXiv e-prints*, 2016.