33rd European Modeling & Simulation Symposium
18th International Multidisciplinary Modeling & Simulation Multiconference

ISSN 2724-0029 ISBN 978-88-85741-57-7 © 2021 The Authors.
doi: 10.46354/i3m.2021.emss.023

Airport passenger flow prediction using simulation data farming and machine learning

Roberto Salvador Félix Patrón[1,*], Paolo Scala[2] , Miguel Mújica Mota[1] and Alejandro Murrieta Mendoza[1]

[1]Aviation Academy, Amsterdam University of Applied Sciences, Weesperzijde 190, Amsterdam, 1097DZ, The Netherlands
[2]Amsterdam School of International Business, Amsterdam University of Applied Sciences, Fraijlemaborg 133, Amsterdam, 1102CV, The Netherlands

[*]Corresponding author. Email address: r.s.felix.patron@hva.nl

Abstract
Passenger flow management is an important issue at many airports around the world. There are high concentrations of passengers arriving and leaving the airport in waves of large volumes in short periods, particularly in big hubs. This might cause congestion in some locations depending on the layout of the terminal building. With a combination of real airport data, as well as synthetic data obtained through an airport simulator, a Long Short-Term Memory Recurrent Neural Network has been implemented to predict the possible trajectories that passengers may travel within the airport depending on user-defined passenger profiles. The aim of this research is to improve passenger flow predictability and situational awareness to make a more efficient use of the airport, that could also positively impact communication with public and private land transport operators.

Keywords: LSTM Recurrent Neural Networks, Artificial Neural Networks, optimization, efficiency

1. Introduction

This paper has been set up through a study case of a large-scale European airport, for which the massive waves of passengers arriving and departing from the airport, particularly during the summer period, may lead to long queues at check-in counters and security control. The issue is extended since this large-scale European airport can only be reached by land transport, for which a prediction of the passenger flow may facilitate communication between airport and public and private land transport operators to improve communication and manage the terminal more efficiently. This paper is part of SESAR under the project Integrated multimodal airport operations for efficient passenger flow management (IMHOTEP), and conducted at the Aviation Academy from the Amsterdam University of Applied Sciences (Mujica Mota, Scala, Herranz, Schultz, & Jimenez, 2020).

Different approaches have been proposed to avoid congestions at airports, most of them focused on strategies to reduce waiting times at different points such as check-in and security control (Alodhaibi, Burdett, & Yarlagadda, 2017; Gatersleben & Weij, 1999; Kalakou & Moura, 2015; Milbredt, Castro, Ayazkhani, & Christ, 2017; Nikoue, Marzouli, Clarke, Feron, & Peters, 2015; Wu & Chen, 2019; S.-Z. Zhao, Ni, Wang, & Gao, 2011). However, in order to have better understanding of where the travelers are going, identify bottlenecks within the terminal or suggest places where to add services such as shops or bathrooms, a passenger trajectory prediction with a complete indication of where each passenger may visit during their airport journey is proposed in this paper.

Passenger flow has been widely studied in the transport industry. Liu and Chen (2017) explained the complexity of passenger trajectories prediction using a land-based transportation problem as a case study. They included a detailed literature review which compared the different type of transportation systems studied in the past years, including the type of data used and the type of methodologies applied. However, as they mentioned in their paper: "passenger flow prediction is a time-series, nonlinear, random and unstable problem, which depends mainly on copious amounts of high quality data and methodologies". Large datasets of historical data and the implementation of deep neural networks, as the aforementioned authors did, can help to get a feasible solution even if it is still a complicated matter to understand passenger behavior.

Out of the different available machine learning techniques, deep neural networks have been proven efficient to model complex non-linear problems (Z. Zhao, Chen, Wu, Chen, & Liu, 2017). To predict a sequence where variable-length inputs and outputs are required, recurrent neural networks (RNN) are often used. These type of networks are "connectionist models with the ability to selectively pass information across sequence steps, while processing sequential data one element at a time" (Lipton, 2015). However, RNN fail to capture long-term evolution, which is a problem that can be solved through a Long Short-Term Memory (LSTM) architecture. LSTM neural networks are able to capture features of a time series with longer time spam (Z. Zhao et al., 2017). In our research paper, a LSTM neural network has been implemented to predict passengers' trajectories in the airport terminal.

To get the most out of neural network architectures, large datasets are required to train the model. For this project, large historical datasets that provide airport information are available. Nevertheless, in order to predict individual passenger trajectories, data farming was used to generate synthetic passenger data through an airport simulator. Data farming within the passenger trajectories problem provides the opportunity to create many different passengers' profiles to have a large number of possible combinations in order to simulate human behavior. Data farming is well suited for exploring the intangibles and nonlinearities that influence decision makers (Barry & Koehler, 2004), in our case, human behavior at the airport terminals.

Simulation models have been successfully used to predict bottlenecks in airports for the last few decades. Gatersleben (1999) applied dynamic modeling in order to understand bottlenecks at Schiphol, one of major European airports. In his project, not only the data obtained through the simulation model was valuable, but the visualization through the simulator was a powerful tool to show the information obtained to all the involved parties.

Improvements in computing time have allowed simulators to increase the complexity of the model. Agent-based simulation models, which allow the users to create individuals with specific characteristics can be of great help to better understand passenger behavior inside the terminal. In a recent study (Verma, Tahlyan, & Bhusari, 2020), an agent-based simulation model was implemented in order to understand passenger behavior, detect the causes of bottlenecks and define a set of policies for improvements. This was made through an analysis based on the simulator data.

In this paper, a combination of real historical data with data farming has been used. The historical data has been provided by a large-scale European airport. First, a deep neural network has been applied to estimate the number of passengers arriving or departing based on information from a given set of flights. This provides an overview of the quantity of people in the terminal within a period of time. Once the estimated amount of people is known, a set of user-defined passenger profiles is used to predict the trajectories that they will be likely to follow once they arrive at the airport. This has been possible through the collection of extensive data created by a detailed airport simulator developed in CAST® (ARC, 2020). This agent-based simulation data provided the inputs to feed a LSTM RNN to predict the trajectories the passengers would follow and with this, understand their behavior and allow the airport crew to be prepared for these big waves of passengers.

2. Methodology

The methodology is divided in three sections: The prediction of the numbers of passengers arriving or departing to the airport (2.1), the data farming process through the airport simulator (2.2) and the passenger trajectory prediction algorithm (2.3).

2.1. Prediction of the number of passengers in a flight

This first step consists in using real historical data at a large-scale European airport. Real data from 1 July 2019 to 30 August 2019 has been used to predict the number of passengers arriving or departing the airport, as well as the gate at which they arrive or depart. These predictions are later fed to the trajectory prediction algorithm. These predictions are intended to be replaced with real data and has been implemented in order to have a better understanding of the passengers at the terminal within a range of time.

Within this period, a total of 41320 flights have been added to the prediction algorithm. Out of these flights, 20648 are departures and 20672 are arrivals. These flights were filtered from the total number of available flights depending on the destination (UE Schengen, UE non-Schengen and National), and only regular passengers' flights with over 50 passengers were included. This number was arbitrarily selected in order to analyze the flights with possible congestion impact.

As this type of data can be considered nonlinear, random and unstable, a Multilayer Perceptron (MLP), which is a type of feed-forward neural network, has been implemented. In Fig. 1 there is an example of an MLP architecture including 6 inputs, 4 hidden layers with 10 neurons each, and 2 outputs. A detailed explanation of the functioning of neural networks is considered to be out of the scope of this paper. The paper by Gardner and Dorling (1998) explains and reviews some applications of MLP networks.

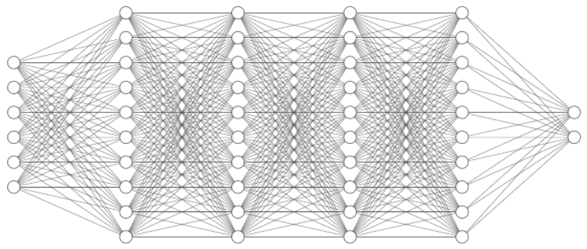
Figure 1 Architecture example of an MLP (LeNail, 2019).

This MLP has the following inputs:

- Month (Jul-Aug)
- Day (Mon-Sun)
- Time of the day: Morning (0-12h), afternoon (12-18h) or night (18-24h)
- Airline
- City (destination or origin)
- Type (arrival or departure)

The outputs are the estimated number of passengers and the gate. The overview of inputs and outputs can be seen in Table 1.

The MLP algorithm has been applied using Scikit-learn (Pedregosa et al., 2011). The 41320 total flights have been randomly split into 80% for training of the algorithm and 20% for testing. The MLP includes 6 input layers, 4 hidden layers with 100 neurons, 2 output layers. The activation functions for the hidden layers are *tanh* and the weight optimization through an *adam* solver. These parameters have been calculated using an exhaustive search method included within Scikin-learn (Pedregosa et al., 2011). The categorical inputs have been one-hot encoded, and inputs and outputs have been scaled.

Table 1 Data sample for the MLP algorithm

Inputs						Outputs	
Month	**Day**	**Time of day**	**Airline**	**City**	**Type**	**Passengers**	**Gate**
7	Mon	Mor.	RYR	STN	departure	168	A36
7	Wed	Mor.	RYR	CGN	departure	155	A26
7	Fri	Aft.	RYR	FRA	departure	162	C45
8	Sat	Aft.	RYR	DTM	departure	176	D81
8	Sat	Night	RYR	NUE	departure	166	A24
7	Tue	Night	RYR	CRL	arrival	190	C48
7	Mon	Night	RYR	MAD	arrival	147	C36
7	Wed	Mor.	RYR	CGN	arrival	121	B36
8	Thu	Mor.	RYR	CGN	arrival	130	D93
8	Mon	Aft.	RYR	CGN	arrival	135	C72

The results of this MLP will be presented in Section 3.1.

2.2. Data farming and the use of synthetic data

Simulation was used for generating synthetic data to be successively used in the machine learning algorithm described in Section 2.3 for generating the passengers' trajectories. The simulation model was built based on an agent-based simulation software CAST® (ARC, 2020), which allowed us to recreate the whole airport terminal including the layout, processes, and the passengers' behavior (Mujica Mota et al., 2020). Some of these processes are exclusive for arrival (A) or for departure (D) passengers, as well as for Schengen and non-Schengen flights. In Table 2 the operations that were simulated in the model are listed.

Table 2 Values of each parameter for building the passengers' profiles.

OPERATION	Passenger type	Passenger status
Check-in	D	Schengen/non-Schengen
Boarding pass scan	D	Schengen/non-Schengen
Security checkpoint	D	Schengen/non-Schengen
Passport control	D	Non-Schengen
Shopping/catering area	D/A	Schengen/non-Schengen
Gate boarding lounge areas	D	Schengen/non-Schengen
Gate boarding desk	D	Schengen/non-Schengen
Baggage claim	A	Schengen/non-Schengen

The synthetic data is a collection of two variables: the passenger location and passenger time stamp at each location. These variables are tracked during the simulated passenger arrival/departure processes. Finally, passenger trajectories are derived by collecting these variables for each of the operations described in Table 2, and they are given as input for the machine learning algorithm. The simulation model was built based on a large-scale European airport, by using real data as input. Figures 2 and 3 show the passengers terminal operations for departures and arrivals flows, respectively.

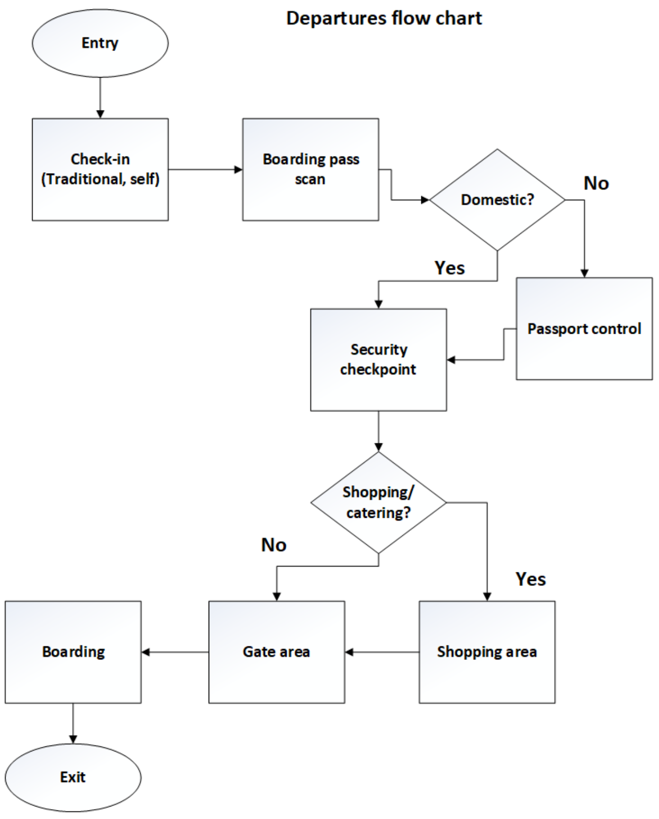

Figure 2. Departure flow operations.

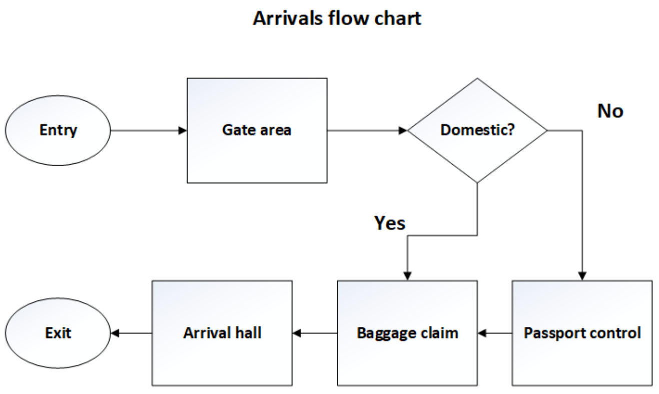

Figure 3. Arrival flow operations.

Different passengers' profiles have been generated as an input of the simulation model and they represent an initial attempt of the authors to characterize the different passengers' trajectories generated by the machine learning algorithm. In total 24 passenger profiles have been generated based on different characteristics arbitrarily chosen by the authors.

The data used as input in the simulation model refers to daily flight schedules of six days during high season (July and August). In this way a large amount of data could be generated to be fed into the passenger trajectory prediction algorithm.

2.3. Passenger trajectory prediction

The passenger trajectory prediction has been created using the data obtained through the data farming process using CAST®(ARC, 2020), and explained in Section 2.2. The dataset includes information about the exact moment a specific passenger enters or exits an object. An object in this context could be a gate, a lounge area or passport control, among others. There is a total of 135 objects in the dataset, 79 gates and the trajectories of 46020 passengers. In order to differentiate the gates from other objects, these values start at 300, as it can be seen in the last element of Table 3 (318 represents gate C72). Objects can be seen in the second column of Table 3. Each object has a unique identifier. For example, Object 28 represents the B_Security object, which has been modeled in our airport simulator as the Security checkpoint B. For this results section, only the passengers in a departure flight have been considered since these typically spend more time at the terminal than passengers arriving from a flight.

Table 3 Data sample for one passenger for the MLP algorithm.

(Unique Object Identifier, Timestamp)	Objects
(70, Timestamp('2019-07-01 03:28:50'))	T2_Checkin
(61, Timestamp('2019-07-01 03:40:15'))	North_Boardingpass
(28, Timestamp('2019-07-01 03:44:10'))	B_Security
(57, Timestamp('2019-07-01 04:02:30'))	ModuleA
(56, Timestamp('2019-07-01 04:03:10'))	Manual_Dep_Passport
(19, Timestamp('2019-07-01 04:11:50'))	A_Gatelounge
(120, Timestamp('2019-07-01 04:50:30'))	T0
(318, Timestamp('2019-07-01 04:50:30'))	GateC72

In Table 3 a sample passenger trajectory is shown, which shows a sequence of variable size for each passenger included in the simulation. This sequence is variable because one passenger may visit more objects that another. An example of a trajectory vector can be derived from Table 3 in the form [70, 61, 28, 57, 56, 19, 120, 318], which represent eight different objects. The original dataset is preprocessed, and a new table is created with *n* rows, which represent the number of passengers. Each row has a variable-size vector with information about the chronologically visited objects. The data preprocessing steps are numbered below:

1. Load all the passenger data obtained from the data farming process
2. Select the visited objects by every passenger
3. Analyze the timestamp and order the sequence chronologically
4. Define the sequence of objects for each passenger

An example of the result of the data preprocessing can be seen in Table 4.

Due to the variety of a passenger's journey through an airport and the amount of possible sequences, a LSTM RNN has been implemented to predict passengers' trajectories. These type of neural networks have an architecture design to be better at storing and accessing information that standard recurrent neural networks (Graves, 2013). The memory from the LSTM architecture can be used to generate complex and realistic sequences containing long-range structures (Graves, 2013).

The LSTM RNN has been implemented to predict a sequence. However, due to the techniques used to optimize the neural network such as the vanishing gradient and exploding gradient problems, the accuracy of the prediction decreases as the sequence gets longer (Z. Zhao et al., 2017). Therefore, fixed-length sequences of six objects have been used.

Using the previous example where the sequence of a passenger is given by [70, 61, 28, 57, 56, 19, 120, 318], if the input of the algorithm is a fixed sequence with six elements, [70, 61, 28, 57, 56, 19], the output of the prediction algorithm should be 120, which is the object that follows in the sequence. The next sequence takes the last 6 elements of the new sequence, giving [61, 28, 57, 56, 19, 120], and the output should be 318.

To summarize the process, the preprocessed data is unrolled into one vector with all the objects from all the passengers, and then these objects are split into input sequences of six objects and outputs of one object. An example for three passengers is shown in Tables 4-6.

Table 4 Sequence sample for three passengers.

Passenger	Sequence
PAX 1	[70, 63, 64, 7, 59, 119, 36, 99, 120, 356]
PAX 2	[70, 61, 62, 4, 74, 59, 128, 43, 36, 106, 99, 120, 356]
PAX 3	[70, 61, 62, 8, 73, 59, 127, 31, 36, 101, 99, 120, 356]

Table 5 Sample of an unrolled sequence.

Unrolled sequence
[70, 63, 64, 7, 59, 119, 36, 99, 120, 356, 70, 61, 62, 4, 74, 59, 128, 43, 36, 106, 99, 120, 356, 70, 61, 62, 8, 73, 59, 127, 31, 36, 101, 99, 120, 356]

Table 6 Sample data as used by the LSTM

Inputs	Output
[70, 63, 64, 7, 59, 119]	36
[99, 120, 356, 70, 61, 62]	4
[74, 59, 128, 43, 36, 106]	99
[120, 356, 70, 61, 62, 8]	73
[59, 127, 31, 36, 101, 99]	120

The objects are treated as categories for the LSTM algorithm, since the objective is to predict an object and not a number. One hot encoding has been applied to these categories. One hot encoding converts integers into a binary matrix, which facilitates the learning process for the neural network.

The LSTM algorithm has been implemented using Keras (Chollet, 2015). All of the data has been used for the training of the algorithm. The model includes an LSTM layer with 256 memory units, a dropout layer with a probability of 20% and a dense layer with a softmax activation. The dropout layer is used to prevent overfitting by randomly dropping out nodes during the training of the model, while the dense layer is used to connect all the neurons of the previous layer to the output layer. A detailed explanation of the functioning of LSTM RNN is considered to be out of the scope of this paper. The paper by Hochreiter and Schmidhuber (1997) explains the theory behind LSTM RNN.

After the model has been trained, the new sequences will be predicted. In Section 2.1 it was mentioned that the information provided was the number of passengers of a flight and the assigned gate. As an example, if this information was Gate A36 and 168 passengers (Table 1), the algorithm will randomly look for a passenger trajectory ending in gate A36 to use it as a start sequence, and then create 168 trajectories from the prediction model. A required step was made to identify the starting and ending points of a sequence. Object 70 provides the moment the passengers arrive to the airport, which means this will always be the start of the sequence. The gate, which numbering starts at 300, indicates the exit point of the passenger. All of the 168 created trajectories in this example will start with the airport arrival and end up in gate A36.

Finally, to generate the new value for each sequence, temperature sampling has been used

(Hinton, Vinyals, & Dean, 2015). This function allows our prediction algorithm to select different sequences with high probability, and not only the sequence with the highest probability.

3. Results

The results section is divided in two: The prediction of the numbers of passengers departing at the airport (3.1) and the passenger trajectory prediction algorithm (3.2).

A flow chart that describes the global algorithm can be seen in Fig. 4. The real airport data feeds the data farming process to generate the passenger trajectories and train the neural network which will generate passenger trajectories. The real airport data is also used to feed the MLP predictor described in Section 3.1, as well as used to generate trajectories in Section 3.2. The results from the MLP predictor are used to generate trajectories as well.

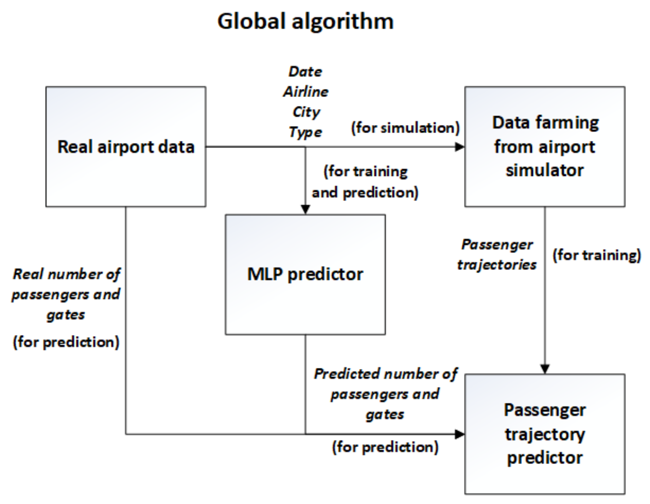
Global algorithm
Date
Airline
City
Type
(for simulation)
Real airport data
Data farming from airport simulator
(for training and prediction)
MLP predictor
Passenger trajectories
(for training)
Real number of passengers and gates
(for prediction)
Predicted number of passengers and gates
(for prediction)
Passenger trajectory predictor

Figure 4. Global algorithm flow chart.

3.1. Prediction of the number of passengers in a flight

The MLP neural network described in Section 2.1 provides a prediction of the number of passengers expected in a flight, either arrival or departure, as well as the expected gate arrival or departure gate. The purpose of this prediction is to have a quick overview of the number of passengers moving inside the airport to use this an input for the passenger trajectory prediction algorithm. These results should be replaced with real schedules if available. Therefore, the accuracy of this algorithm is not critical for the purpose of this research project.

The number of passengers' prediction has a R^2 score of **59.6%**. The R^2 score, or coefficient of determination, shows how fit an algorithm is to make predictions. This is calculated as follows (Pedregosa et al., 2011):

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

Where $\hat{y}$ is the predicted value.

The mean absolute error of the algorithm is **15.3** (number of passengers). The mean absolute error has been calculated with Eq. 2.

$$MAE(y,\hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i| \qquad (2)$$

These results show the difficulty the neural network has with the nonlinearity and randomness of some of the flight data. However, this prediction is only used as a base for the trajectory prediction algorithm, which allows this result to be acceptable.

The second output of the MLP algorithm predicts the departure of arrival gate. When calculating the terminal, and not focusing in the actual gate number, the algorithm gives an accuracy of **90.9%**. This means that the correct terminal is predicted accurately.

3.2. Passenger trajectory prediction

The results in this section consider only departure passengers. This choice was made arbitrarily, based on the assumption that passengers on departure may spend more time in the shopping and diner areas than arrival passengers. As it was discussed before, the passenger trajectory prediction algorithm takes as input the number of passengers and the departure gate. Two different sets of data are discussed below: real airport information from 7 July 2019 considering flights between 18h and 19h, and predicted data from the algorithm discussed in section 3.1 using as inputs the same information as in the real data (see Table 1), but instead predicting the number of passengers and departure gates. The date selected is a Sunday in the summer season from a holiday destination airport, which would mean that a high number of departing passengers would be in the terminal.

The input data for the trajectory prediction algorithm can be seen in Table 7.

As it can be seen, the number of passengers and gates predicted are in accordance with the results discussed in Section 3.1.

In total, 3608 trajectories have been created using the real scheduled data, and 3575 using the predicted data from the algorithm described in Section 3.1. In Table 8, the eight most visited objects can be seen. This can allow the staff at the terminal to visualize where the congested points will be.

Table 7 Input data for the passenger trajectory prediction algorithm

General information				Passengers		Gates	
Date	**Time**	**Airline**	**City**	**Real**	**Pred.**	**Real**	**Pred.**
7-Jul-19	18:00	EJU	SXF	153	142	C63	C40
7-Jul-19	18:05	LDM	DUS	168	186	C57	C46
7-Jul-19	18:10	IBK	CPH	181	178	D90	C52
7-Jul-19	18:10	LDM	DUS	171	186	C44	C46
7-Jul-19	18:15	SWR	ZRH	167	171	D86	C41
7-Jul-19	18:20	LDM	DUS	173	186	C72	C46
7-Jul-19	18:20	CFG	DUS	272	248	C54	C50
7-Jul-19	18:20	RYR	CIA	179	180	C38	C50
7-Jul-19	18:25	NAX	OSL	179	168	D84	C43
7-Jul-19	18:25	VOE	TLS	147	157	C48	C56
7-Jul-19	18:35	EXS	NCL	185	179	A16	A18
7-Jul-19	18:35	EWG	STR	166	155	C64	C43
7-Jul-19	18:40	LDM	STR	172	174	C52	C46
7-Jul-19	18:40	RYR	NRN	174	178	C50	C46
7-Jul-19	18:45	LDM	VIE	192	174	C46	C45
7-Jul-19	18:50	RYR	FKB	186	178	C46	C39
7-Jul-19	18:55	RYR	BCN	186	188	C40	C46
7-Jul-19	18:55	LGL	LUX	161	167	D92	C46
7-Jul-19	18:55	CFG	LEJ	219	211	C48	C45
7-Jul-19	19:00	EXS	GLA	177	169	A10	A08

Table 8 Number of passengers per object in the created trajectories.

Object name	Passengers	
	Real	**Pred.**
['TPerson.entryTime__T2__Checkin [Date/Time]'	3608	3575
['TPerson.entryTime__ModuleC [Date/Time]'	2395	2671
['TPerson.entryTime__C__Gatelounge [Date/Time]'	2294	2571
['TPerson.entryTime__C__gate [Date/Time]'	2225	2498
['TPerson.entryTime__South__Security [Date/Time]'	1805	1747
['TPerson.entryTime__South__Boardingpass [Date/Time]'	1805	1747
['TPerson.entryTime__North__Security [Date/Time]'	1674	1746
['TPerson.entryTime__North__Boardingpass [Date/Time]'	1674	1746

To better visualize the results from Table 8, Figures 5 and 6 have been created. These have been color coded in order to show in red the most congested points. In this example, the main congestion point would be at the check-in area, as it would be expected, and can be seen in Fig. 5. In Figure 6, the security areas are shown.

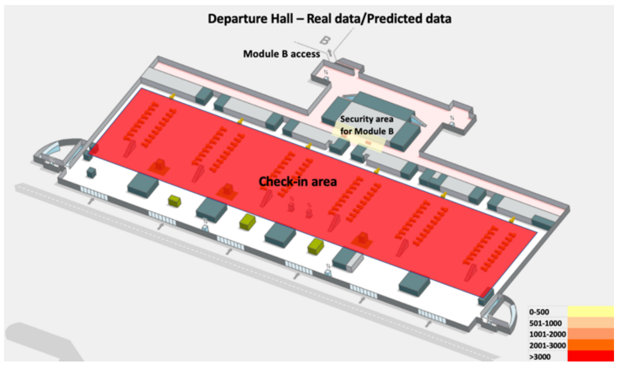
Departure Hall – Real data/Predicted data
Module B access
B
Security area for Module B
Check-in area
0-500
501-1000
1001-2000
2001-3000
>3000

Figure 5. Departure hall congestion visualization on 7/7/19 from 18:00-19:00. Base image source: (AENA, 2021).

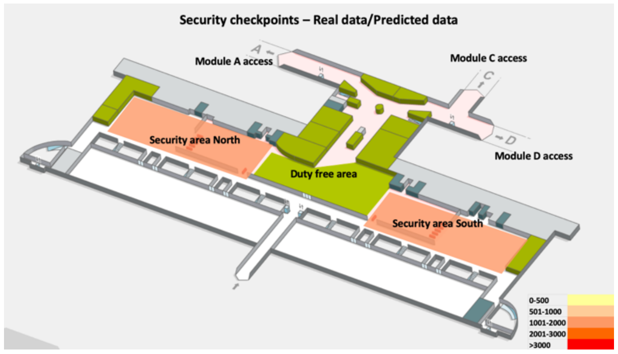

Figure 6. Security checkpoints congestion visualization on 7/7/19 from 18:00-19:00. Base image source: (AENA, 2021).

4. Conclusions

The proposed trajectory prediction algorithm uses a combination of real data, data farming and machine learning techniques in order to predict the most visited areas at an airport terminal for a given period. This tool could be used to predict congestion points. However, the passenger profiles through data farming could be used to better understand the behavior of the different types of passengers at airports. These profiles, if defined using real consumer data, for example, could give the airport an overview of the type of shops, dining options, gate lounges, etc., that would be better suited for these specific profiles.

References

AENA. (2021). AENA. Retrieved from https://portal.aena.es/csee/Satellite?Language=EN_GB&ca=PMI&pagename=cartografia&ps=t&ti=T

Alodhaibi, S., Burdett, R. L., & Yarlagadda, P. K. D. V. (2017). Framework for Airport Outbound

Passenger Flow Modelling. *Procedia Engineering, 174*, 1100-1109. doi:https://doi.org/10.1016/j.proeng.2017.01.263

ARC. (2020). CAST. Retrieved from https://arc.de/cast-simulation-software/

Barry, P., & Koehler, M. (2004, 5-8 Dec. 2004). *Simulation in context; using data farming for decision support.* Paper presented at the Proceedings of the 2004 Winter Simulation Conference, 2004.

Chollet, F. (2015). Keras. *GitHub repository*. Retrieved from https://github.com/fchollet/keras

Gardner, M. W., & Dorling, S. R. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment, 32*(14), 2627-2636. doi:https://doi.org/10.1016/S1352-2310(97)00447-0

Gatersleben, M. R., & Weij, S. W. V. d. (1999, 5-8 Dec. 1999). *Analysis and simulation of passenger flows in an airport terminal.* Paper presented at the WSC'99. 1999 Winter Simulation Conference Proceedings. 'Simulation - A Bridge to the Future' (Cat. No.99CH37038).

Graves, A. (2013). Generating Sequences With Recurrent Neural Networks. *ArXiv, abs/1308.0850*.

Hinton, G. E., Vinyals, O., & Dean, J. (2015). Distilling the Knowledge in a Neural Network. *ArXiv, abs/1503.02531*.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*(8), 1735-1780. doi:10.1162/neco.1997.9.8.1735

Kalakou, S., & Moura, F. (2015). Modelling Passengers' Activity Choice in Airport Terminal before the Security Checkpoint: The Case of Portela Airport in Lisbon. *Transportation Research Procedia, 10*, 881-890. doi:https://doi.org/10.1016/j.trpro.2015.09.041

LeNail, A. (2019). NN-SVG: Publication-Ready Neural Network Architecture Schematics. *J. Open Source Softw., 4*, 747.

Lipton, Z. C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning. *ArXiv, abs/1506.00019*.

Liu, L., & Chen, R.-C. (2017). A novel passenger flow prediction model using deep learning methods. *Transportation Research Part C: Emerging Technologies, 84*, 74-91. doi:https://doi.org/10.1016/j.trc.2017.08.001

Milbredt, O., Castro, A., Ayazkhani, A., & Christ, T. (2017). Passenger-centric airport management via new terminal interior design concepts. *Transportation Research Procedia, 27*, 1235-1241. doi:https://doi.org/10.1016/j.trpro.2017.12.008

Mujica Mota, M., Scala, P., Herranz, R., Schultz, M., & Jimenez, E. (2020). *Creating the future airport passenger experience: IMHOTEP.* Paper presented at the Proceedings of the 32nd European Modeling & Simulation Symposium (EMSS 2020).

Nikoue, H., Marzouli, A., Clarke, J. P., Feron, E., & Peters, J. (2015). *Passenger Flow Predictions at Sydney International Airport: A Data-Driven Queuing Approach.* Retrieved from

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res., 12*(null), 2825–2830.

Verma, A., Tahlyan, D., & Bhusari, S. (2020). Agent based simulation model for improving passenger service time at Bangalore airport. *Case Studies on Transport Policy, 8*(1), 85-93. doi:https://doi.org/10.1016/j.cstp.2018.03.001

Wu, C.-L., & Chen, Y. (2019). Effects of passenger characteristics and terminal layout on airport retail revenue: an agent-based simulation approach. *Transportation Planning and Technology, 42*(2), 167-186. doi:10.1080/03081060.2019.1565163

Zhao, S.-Z., Ni, T.-H., Wang, Y., & Gao, X.-T. (2011). A new approach to the prediction of passenger flow in a transit system. *Computers & Mathematics with Applications, 61*(8), 1968-1974. doi:https://doi.org/10.1016/j.camwa.2010.08.023

Zhao, Z., Chen, W., Wu, X., Chen, P. C. Y., & Liu, J. (2017). LSTM network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems, 11*(2), 68-75. Retrieved from https://digital-library.theiet.org/content/journals/10.1049/iet-its.2016.0208