CS230

Prediction of the Estimated Time of Arrival of an airborne aircraft using Deep Learning

Guillermo Frontera Sánchez
frontera@stanford.edu

Marta Palomar Toledano
mpalomar@stanford.edu

1 Introduction

The problem we aim to solve is the prediction of the Estimated Time of Arrival (ETA) of an airborne aircraft. That is, given an aircraft that has already taken off, we want to be able to provide an estimation of how long it will take that aircraft to land at its destination airport.

Having an accurate prediction of the Estimated Time of Arrival (ETA) of an aircraft during flight is an issue of great interest for passengers, airlines, airspace and airport operators, and other stakeholders. An accurate prediction of the ETA is crucial in order to speed up runway, management and terminal operations, as well as increase passenger's service quality.

Many of the interesting applications of this estimation would benefit from knowing not only the time at which the aircraft will land, but also the time at which the aircraft will reach the corresponding gate at the airport. Our model also provides predictions of the time of arrival at the gate.

This prediction is a complex process depending on several factors including the aircraft's own flight dynamics, airspace congestion, and environmental conditions. It is relevant to highlight that those types of affecting factors do suffer from variability (i.e., weather conditions might change during flight, as well as potential conflicts with other traffic) and therefore are highly non-deterministic.

We used a multi-task learning solution using a deep neural network to simultaneously predict the landing time and gate arrival time.

2 Related work

Pioneer works aiming to solve the ETA estimation problem addressed it by employing mainly deterministic approaches which mostly rely on aircraft performance models. Nonetheless, given the highly non-deterministic behavior of the input variables involved in the ETA estimation problem, more recent works have been applying different data-driven techniques. For instance, in Ayhan et al. [2], various regression models and a Recurrent Neural Network (RNN) are employed. Basturk and Cetek [1] recently published a work covering this topic of ETA prediction based on Machine Learning.

3 Dataset

A significant amount of effort has been spent on the creation of the dataset that we used to train our model. We had to compose the dataset from various sources, including air traffic data, weather data, flight plan data, and aircraft performance data.

CS230: Deep Learning, Fall 2021, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

3.1 Data retrieval

Our main source of information is a database of collected traffic information provided by FlightRadar24[1]. This database contains worldwide information of air traffic, where each record corresponds to information of one flight at one point in time. Two example records are provided in Table 1. For each flight in the database, a sample is available every 5 to 10 seconds, although this frequency may vary in some cases.

Table 1: Sample FlightRadar24 data

Time stamp	Hex. Id	Call sign	Latitude	Longitude	Altitude	Ground speed	Track	Vertical rate	Aircraft type	On ground	Origin	Destination
2019-03-21 05:32:08	A2EBCA	AAL740	40.37820	−5.39240	33450	445	116	−2048	A332	0	PHL	MAD
2021-10-10 01:22:12	AC297B	FDX6014	39.70291	−86.29346	0	0	337	0	B77L	1	ANC	IND

The first task that we have done is computing the 'ground truth' for our arrival times, both for the landing on the runway and parking at the gate. For this, we have re-used an algorithm that we developed in the past, which takes the sequence of records for a flight and a map of the airport (for obtaining the positions of the runways and the parking areas at the gates) and computes both arrival times. This information was generated once and stored in a database that we will refer to as the arrivals database. Example records for this database are shown in Table 2.

Table 2: Sample arrivals data

Airport	Hex. Id	Call sign	Landing time	Gate time	Aircraft type	Runway	Parking area
EHAM	4BB853	PGT69B	2019-01-01 01:07:51	2019-01-01 01:21:49	A20N	24	G7
LEMD	34520F	IBE6341	2021-01-01 02:38:17	2021-01-01 02:43:13	A332	36L	536

Next, we wanted to obtain a measurable method to quantify the congestion at an airport at any given time. We found that good way of getting an approximate measure of this congestion was counting the number of scheduled departures and arrivals for one airport during a determined time interval. For this, we took a database of flight plans and we counted the number of flights scheduled to take off or land at each airport during any 30-minute interval. This information is stored in a database that we will refer to as the congestion database, for which a few examples are provided in Table 3.

Table 3: Sample congestion data

Airport	From time	To time	# departures	# arrivals
LEMD	2019-03-01 16:00:00	2019-03-01 16:30:00	11	16
LEMD	2019-03-01 16:30:00	2019-03-01 17:00:00	10	9

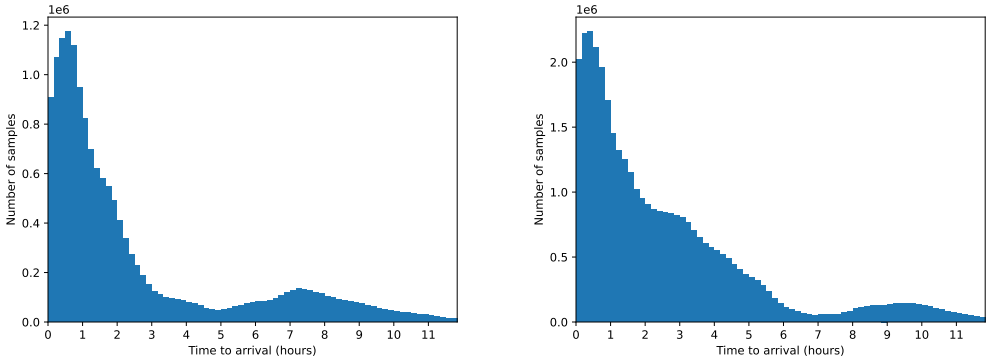
Additionally, we collected the relevant METeorological Aerodrome Reports (METARs), which contain meteorological observations at airports and are issued at regular 30-minute intervals. Each record includes the airport and time corresponding to the observation, the temperature, humidity, wind direction and speed, visibility, humidity, pressure, presence of clouds (including altitude and type of clouds for up to four layers), precipitation, etc. This weather database also contains the reference latitude and longitude for each airport, which became useful at a later step in the process.

And finally, we have obtained a database of aircraft performance information. This database contains, for each aircraft model, information such as the type and number of engines, length, wingspan, minimum and maximum weight, and length requirements on the runway for take-off and landing.

The dataset could then be generated. Each record in the traffic database is combined with the arrival information for that flight (using the call sign and time stamp to make the correspondence), the weather information for the destination airport at the time stamp, the congestion information for the destination airport and the time slot prior to the time stamp, and the aircraft model information.

We have generated two separate datasets, each containing data for a different airport. We have chosen Madrid-Barajas Airport (MAD) and San Francisco International Airport (SFO). Each dataset contains data corresponding to 20 months of flights, from March 2019 to December 2020. To make the dataset smaller (using all the data available was non-viable due to its huge size), we randomly discarded 95% of the examples. Despite the reduction in size, the dataset for Madrid-Barajas Airport

[1]FlightRadar24 (`https://www.flightradar24.com/`) is a service that collects location information that most commercial aircraft (including airliners and cargo aircraft) broadcast via radio signals using a technology known as Automatic Dependent Surveillance – Broadcast (ADS-B).



(a) MAD (b) SFO

Figure 1: Histogram of samples in each dataset by the time to arrival.

contains approximately 15.8 million records (about 5.2 GB of data), and the dataset for San Francisco International Airport contains approximately 36.8 million records (about 12 GB of data). Fig. 1 contains the distribution of the records by their time to arrival in both datasets, showing that we have many more examples of aircraft that are close to landing than we have of aircraft that are 4 or more hours away from landing.

3.2 Data pre-processing

The dataset contains a lot of information that could be used to make a good prediction of the ETA. However, each record contains categorical data (for example, the call signs) and potentially missing data (e.g., METARs provide many 'blank' features when not available or not applicable). There is also some information that we can compute that is highly correlated to what we try to predict. To solve these problems and improve the data, we pre-process the dataset prior to feeding it to the model.

For categorical data, we assign a numeric encoding to each category. In most cases, we use one-hot encoding; but in some cases where it makes sense, we manually assign a number to each category (for example, in METARs, cloud coverage uses categories that imply what fraction of the sky is occupied by clouds, so we replace the category by the actual fraction). Also, for the call sign, we want to take the first three characters, which identify the airline, and use them as an additional category. For missing numeric values, we use a simple imputation transformer to fill in the blanks with the median or a constant value. The time stamp of each record is broken down in several features: year, month, day, time of the day, day of the year, and day of the week. We also compute the geodesic distance between the aircraft and the destination airport for each record.

Finally, we standardize all features by subtracting the mean and scaling to a variance of one.

4 Learning Method

A Dense Neural Network (DNN) architecture has been chosen to handle this prediction problem. It consists of a set of fully connected layers with ReLUs as activation functions and an output layer composed by two nodes, one for landing time estimation and another for gate time (see scheme in Fig. 2).

Several parameters have been tuned in order to find an optimum, such as the number of layers, the number of nodes within each layer, the learning rate and activation functions. The values that resulted in more accurate predictions are the summarized in Table 4. Keeping these parameters fixed, we trained 4 different model configurations, with 6, 12, 24 and 48 layers consisting of 100 neurons each. Such parametric study was conducted using the two datasets presented in the previous section.

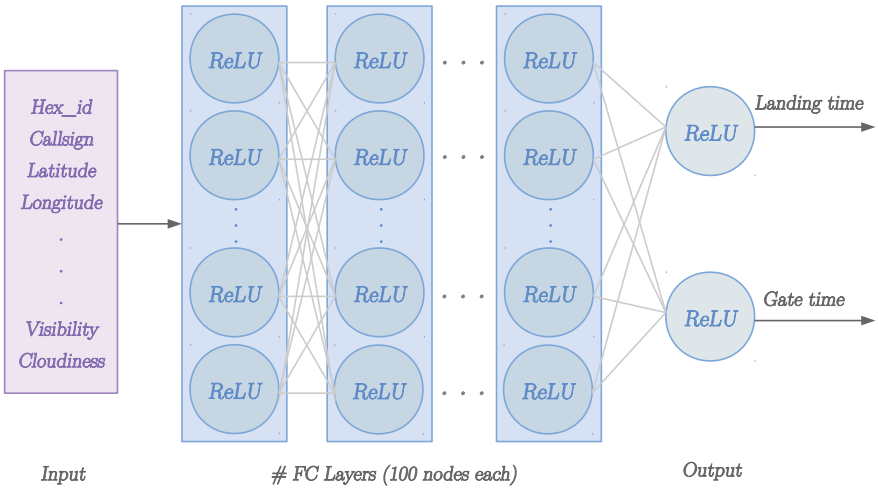


Figure 2: Dense Neural Network architecture

However, the limits in our hardware resources[2] didn't allow us to use the whole datasets as were described in the previous section. Therefore, from each of the datasets we took 5 million random samples that were split into a train set (95%) and a development set (5%).

Table 4: Model parameters

Learning rate	Batch size	L2 parameter	Loss	Optimizer
0.001	512	0.0001	MAE	Adam

The two main performance metrics considered for this problem are the MAE and MAPE. Both of them were assessed as loss functions given their suitability for this specific problem, and best metric values were obtained when optimizing on MAE. For this reason we chose this loss type for subsequent calculations.

Table 5 gathers a summary of some of the computed performance metrics achieved for each configuration with both MAD and SFO datasets. The whole set of metrics assessed in this project for each number of layers and destination airport are collected in the Appendix of this document. Despite the fact that all metrics have the same order of magnitude, it appears that lower error levels were achieved when setting the number of layers at 12 and therefore this configuration for the DNN model will be employed for further comparisons.

Table 5: Performance metrics for the different model configurations

		MAD			SFO		
# Layers	Set	EVS	MAE	MAPE	EVS	MAE	MAPE
6	Train	0.917	342	6.5%	0.979	305	6.1%
	Dev	0.911	361	6.9%	0.975	319	6.3%
12	Train	0.918	328	5.6%	0.981	277	5.6%
	Dev	0.924	321	5.8%	0.978	289	5.8%
24	Train	0.917	384	6.5%	0.982	280	5.8%
	Dev	0.911	372	6.7%	0.977	293	5.8%
48	Train	0.917	378	6.3%	0.978	286	5.9%
	Dev	0.911	361	6.3%	0.973	299	5.9%

[2]We were unable to use P instances in AWS because our limit increase request was denied, even after several disputes.

5 Evaluation

The DNN model developed in this project will be compared with other common strategies employed to solve ETA prediction problems. Specifically, we chose for comparison a ML model based on Random Forests. With the latter, the performance metrics for both the train and development sets are summarized in Table 15 and 16 for MAD and SFO airports, respectively. In order to ease the comparison between models, Table 6 gathers three of the most illustrative metrics for the two datasets. Values for the DNN model correspond to the 12-layer approach using the development set.

Paying attention to the overall metrics, our DNN model results in a better prediction accuracy than using ML. Furthermore, these values should be interpreted taking into account the distribution of the error with the remaining time to arrival: this means that for this specific problem we may want our error to be proportional to the time to arrival. The rationale is that for ATC or pilots to take actions to avoid conflict or congestion at the arrival airport it is desirable to have a more accurate prediction when the aircraft is closer to the touchdown point.

Fig. 5 illustrates the error in the prediction of the landing and gate times for the 6-layer DNN model with respect to the remaining time to arrival on both the test and dev sets. The overall decrease of the error magnitude (in seconds) with the decrease of time to arrival is noticeable, ranging from approximately 8 minutes when the aircraft is 12 hours away from the destination airport to 2 minutes or less when there is half an hour left. On the other hand, if representing this same distribution with the ML approach a different tendency is observed (Fig. 11). The error remains almost constant regardless the actual time to arrival with a mean value of approximately 4 minutes except for the last hour of flight, where error increases up to 10-15 minutes. In view of these results, we can conclude that our DL approach does outperform the one based on ML if taking into account what our interests are (maintaining a decreasing error magnitude as the flight reaches its end).

Note: For predictions using the dataset for the SFO airport, the error versus time to arrival curve is less smooth than in the case of MAD airport, revealing an increase in MAE in the time to arrival interval from 6 to 8 hours approximately. This local rise in error has not been considered to be critical to our final purpose as it occurs when we are reasonably far in time from the touchdown. A possible reason to explain such behavior may be the lower number of samples available for that specific interval (as it can be seen in the histogram of Fig. 1).

Table 6: Metric comparison between the DL and ML models

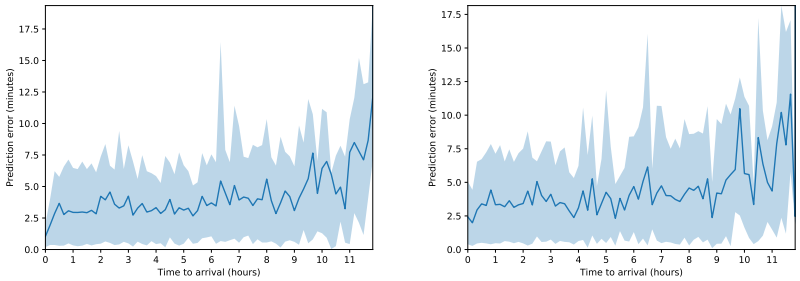
	MAD		SFO	
Metric	DNN model	ML (Random Forests)	DNN model	ML (Random Forests)
EVS	0.924	0.952	0.978	0.985
MAE	321	371	289	302
MAPE	5.8%	54.0%	5.8%	18.2%

Taking the presented deep learning approach we have proved to increase the accuracy of ETA prediction with respect to other methodologies such as ML, making it suitable for some potential applications like air traffic management. However, these results could be refined in future works by employing other DL architectures such as RNNs (specifically LSTM) in conjunction, for instance, with convolutional layers for the spatial and temporal inputs. Other input parameters such as weather or vehicle characteristics could be then concatenated to the spatiotemporal output and passed through a set of fully connected layers. Some other improvements could be achieved by making some improvements in the dataset, such as making the distribution of samples more homogeneous across different times to arrival.

References

[1] Basturk, O. & Cetek, C. (2021). Prediction of aircraft estimated time of arrival using machine learning methods. *The Aeronautical Journal* **1289**(125):1245-1259.

[2] Ayhan, S., Costas, P., & Samet, H. (2018). Predicting Estimated Time of Arrival for Commercial Flights. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*: 33–42.



(a) Landing time error (dev set) (b) Gate time error (dev set)

Figure 3: Error for the predictions based on the 6-layer DNN model for MAD.

[3] Wang, D., Zhang, J., Cao, W., Li, J., & Zheng, Y. (2018). When Will You Arrive? Estimating Travel Time Based on Deep Neural Networks. *AAAI*.

[4] Zhang, K., Lai, Y., Jiang, L., & Yang, F. (2020). Bus Travel-Time Prediction Based on Deep Spatio-Temporal Model. *WISE*.

Appendix: Result Data

This appendix shows the result data obtained using different models.

For each trained model, we show a table containing the Mean Squared Error (MSE), the Root Mean Squared Error (RMSE), the Mean Absolute Error (MAE), the Explained Variance Score (EVS), the Mean Absolute Percentage Error (MAPE), and the Mean Bias Error (MBE). This metrics are provided for both the train set and the test set.

We also provide plots showing how the error varies with the time to arrival. These plots show a line representing the average absolute error, and a shaded area that contains the 10th to the 90th percentiles.

5.1 6 Layer Deep Neural Network

The results obtained after training and predicting arrival times for the Madrid-Barajas Airport (MAD) using a 6-layer DNN are summarized in Table 7 and Fig. 3. The results of using this same model for the San Francisco International Airport (SFO) are summarized in Table 8 and Fig. 4.

Table 7: Results for the predictions based on the 6-layer DNN model for MAD.

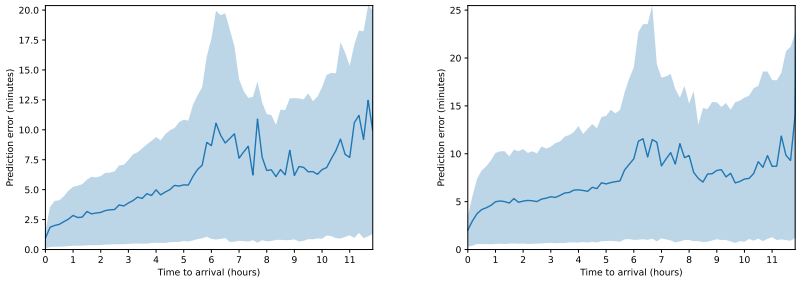
Set	MSE	RMSE	MAE	EVS	MAPE	MBE
Train	10775955	3282	342	0.917	6.5%	188
Dev	11722600	3423	361	0.911	6.9%	190

Table 8: Results for the predictions based on the 6-layer DNN model for SFO.

Set	MSE	RMSE	MAE	EVS	MAPE	MBE
Train	2497544	1580	305	0.979	6.1%	48
Dev	3086185	1757	319	0.975	6.3%	51

5.2 12 Layer Deep Neural Network

The results obtained after training and predicting arrival times for the Madrid-Barajas Airport (MAD) using a 12-layer DNN are summarized in Table 9 and Fig. 5. The results of using this same model for the San Francisco International Airport (SFO) are summarized in Table 10 and Fig. 6.



(a) Landing time error (dev set) (b) Gate time error (dev set)

Figure 4: Error for the predictions based on the 6-layer DNN model for SFO.

Table 9: Results for the predictions based on the 12-layer DNN model for MAD.

Set	MSE	RMSE	MAE	EVS	MAPE	MBE
Train	10739207	3277	328	0.918	5.6%	61.3
Dev	9847404	3138	321	0.924	5.8%	49.5

Table 10: Results for the predictions based on the 12-layer DNN model for SFO.

Set	MSE	RMSE	MAE	EVS	MAPE	MBE
Train	2218562	1489	277	0.981	5.6%	79.1
Dev	2750988	1659	289	0.978	5.8%	82.5

5.3 24 Layer Deep Neural Network

The results obtained after training and predicting arrival times for the Madrid-Barajas Airport (MAD) using a 24-layer DNN are summarized Table 11 and Fig. 7. The results of using this same model for the San Francisco International Airport (SFO) are summarized in Table 12 and Fig. 8.

Table 11: Results for the predictions based on the 24-layer DNN model for MAD.

Set	MSE	RMSE	MAE	EVS	MAPE	MBE
Train	12915522	3594	384	0.917	6.5%	129
Dev	11577045	3402	372	0.911	6.7%	111

Table 12: Results for the predictions based on the 24-layer DNN model for SFO.

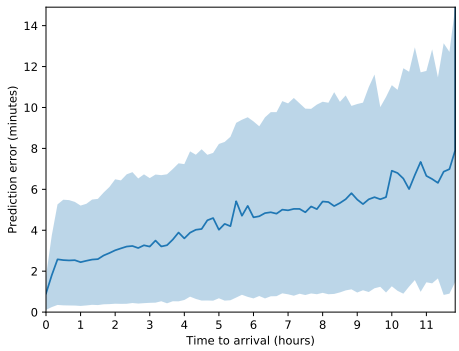
Set	MSE	RMSE	MAE	EVS	MAPE	MBE
Train	2198569	1483	280	0.982	5.8%	65
Dev	2822521	1680	293	0.977	5.8%	70

5.4 48 Layer Deep Neural Network

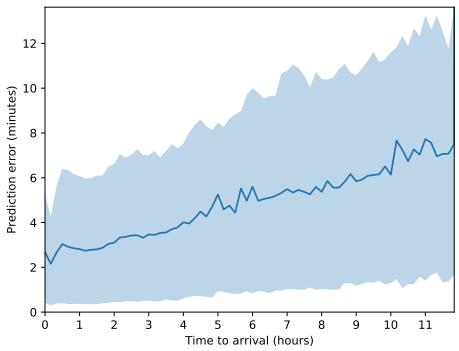
The results obtained after training and predicting arrival times for the Madrid-Barajas Airport (MAD) using a 48-layer DNN are summarized Table 13 and Fig. 9. The results of using this same model for the San Francisco International Airport (SFO) are summarized in Table 14 and Fig. 10.

5.5 Random Forest

The results obtained after training and predicting arrival times for the Madrid-Barajas Airport (MAD) using a Random Forest model are summarized Table 15 and Fig. 11. The results of using this same model for the San Francisco International Airport (SFO) are summarized in Table 16 and Fig. 12.

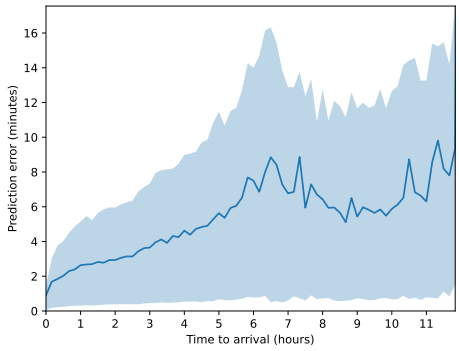


(a) Landing time error (dev set)

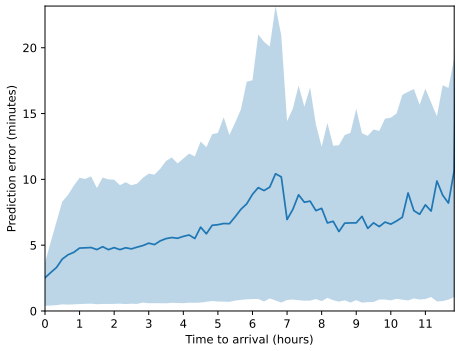


(b) Gate time error (dev set)

Figure 5: Error for the predictions based on the 12-layer DNN model for MAD.



(a) Landing time error (dev set)



(b) Gate time error (dev set)

Figure 6: Error for the predictions based on the 12-layer DNN model for SFO.

Table 13: Results for the predictions based on the 48-layer DNN model for MAD.

Set	MSE	RMSE	MAE	EVS	MAPE	MBE
Train	14036172	3746	378	0.917	6.3%	244
Dev	12379721	3518	361	0.911	6.3%	223

Table 14: Results for the predictions based on the 48-layer DNN model for SFO.

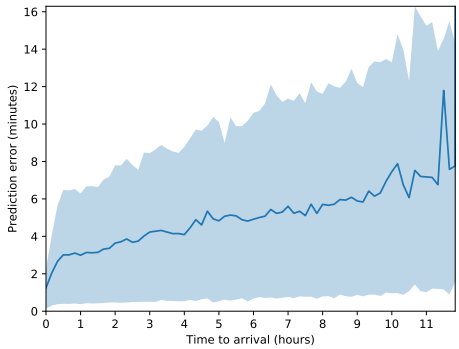
Set	MSE	RMSE	MAE	EVS	MAPE	MBE
Train	2689742	1640	286	0.978	5.9%	62
Dev	3257388	1804	299	0.973	5.9%	66

Table 15: Results for the predictions based on the Random Forest model for MAD.

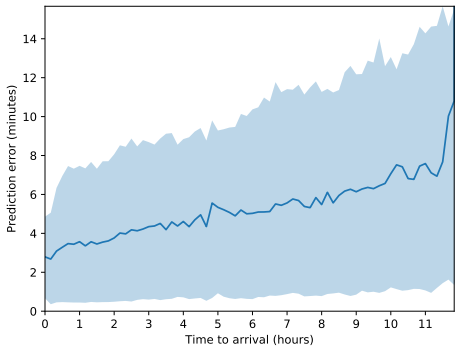
Set	MSE	RMSE	MAE	EVS	MAPE	MBE
Train	4258754	2064	300	0.967	35%	3.87
Dev	6254291	2501	371	0.952	54%	−6.53

Table 16: Results for the predictions based on the Random Forest model for SFO.

Set	MSE	RMSE	MAE	EVS	MAPE	MBE
Train	639603	800	239	0.994	10.4%	1.63
Dev	1730195	1315	302	0.985	18.2%	5.28

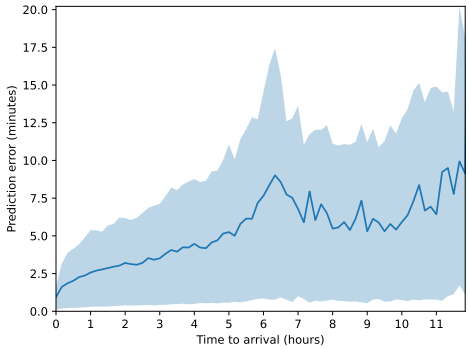


(a) Landing time error (dev set)

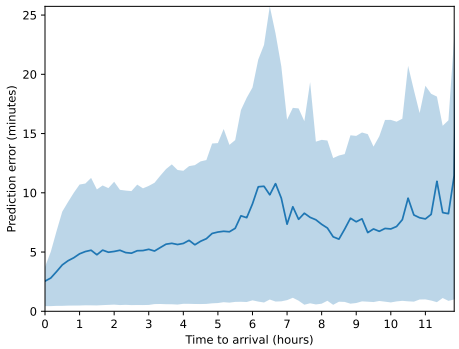


(b) Gate time error (dev set)

Figure 7: Error for the predictions based on the 24-layer DNN model for MAD.

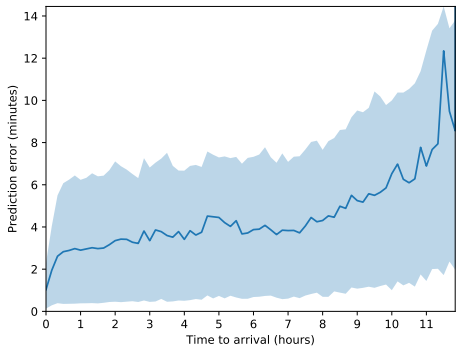


(a) Landing time error (dev set)

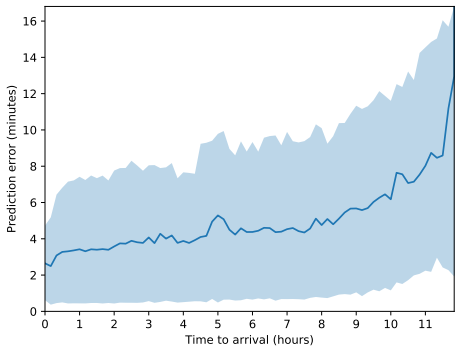


(b) Gate time error (dev set)

Figure 8: Error for the predictions based on the 24-layer DNN model for SFO.

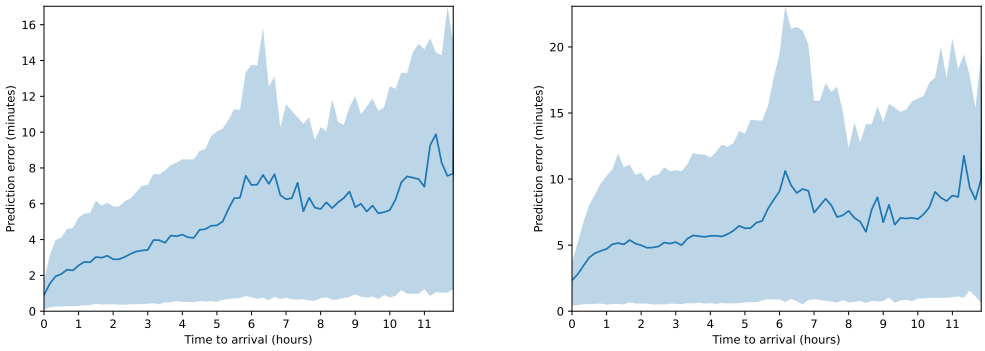


(a) Landing time error (dev set)



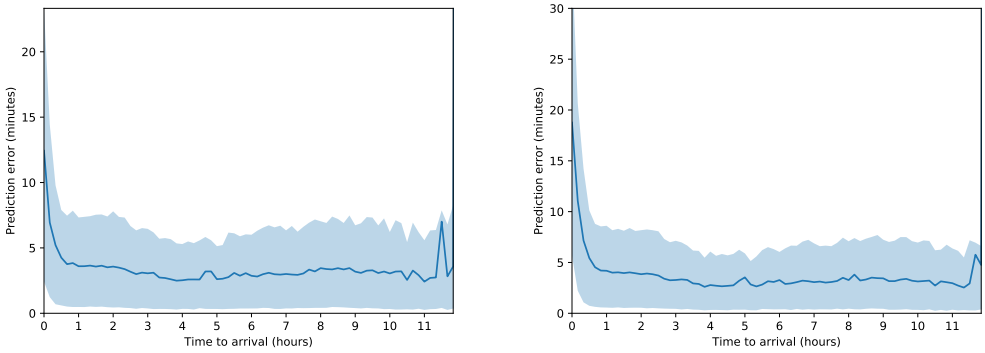
(b) Gate time error (dev set)

Figure 9: Error for the predictions based on the 48-layer DNN model for MAD.



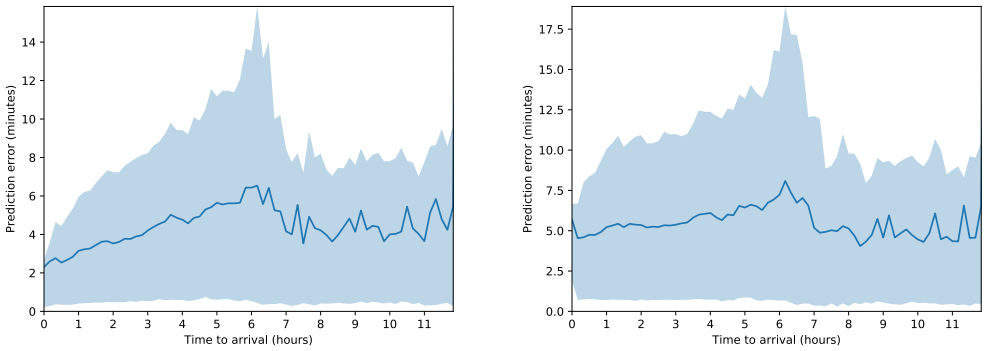
(a) Landing time error (dev set) (b) Gate time error (dev set)

Figure 10: Error for the predictions based on the 48-layer DNN model for SFO.



(a) Landing time error (dev set) (b) Gate time error (dev set)

Figure 11: Error for the predictions based on the Random Forest model for MAD.



(a) Landing time error (dev set) (b) Gate time error (dev set)

Figure 12: Error for the predictions based on the Random Forest model for SFO.