Real-Time Prediction of Go-Around Probability During Final Approach with Deep Learning

Unai Zuazo, Antonio Jiménez-Martín, Alfonso Mateos
Decision Analysis and Statistics Group,
Universidad Politécnica de Madrid (UPM),
Madrid, Spain

Manuel Angel Amaro Carmona
Centro de Referencia de Investigación,
Desarrollo e Innovación ATM A.I.E. (CRIDA),
Madrid, Spain

***Abstract*—This paper presents a real-time system that estimates the probability of a go-around (GA) during final approach using deep learning. Historical approach data from Barcelona–El Prat and Madrid–Barajas — including alignment with the runway, range and descent geometry, aircraft energy state, and in-trail spacing — are fused with meteorological variables from Open-Meteo to train three neural architectures: a Long Short-Term Memory (LSTM) network, an attention-augmented LSTM, and a Transformer. Performance is assessed with distance-to-ideal, temporal accuracy, area under the accuracy curve, and confusion-matrix metrics. Model explainability is provided via SHAP. The Transformer consistently delivers the best results, achieving $>80\%$ accuracy up to two minutes before a GA and $\sim 87\%$ at 30 seconds, and has been integrated into a real-time prototype (LandIA) designed to support air traffic controllers.**

***Keywords*—Go-Around, Real-Time Prediction, Deep Learning.**

I. Introduction

Air traffic management is a complex and dynamic system that requires continuous monitoring and precise decision-making to guarantee both the safety and efficiency of flight operations. One of the most critical phases of a flight is the final approach and landing, where pilots and air traffic controllers (ATCOs) work together to ensure a stable descent. However, landings do not always unfold as expected. In certain situations, pilots must perform a go-around (GA)—an aborted landing maneuver—due to unstable approach conditions, external disturbances, or safety concerns.

Although GAs are standard safety procedures, they pose significant operational challenges [1]. They require immediate decision-making, rapid flight path changes, and careful coordination to prevent conflicts with surrounding traffic. Although they are rare occurrences (around 0.2% of approaches), their unpredictability can increase airspace congestion, delay subsequent operations, and raise fuel consumption. Consequently, the ability to predict go-arounds in real time could enhance situational awareness, reduce controller workload, and strengthen overall safety.

Previous research has examined both environmental and behavioral factors behind GAs:

- *Environmental modeling:* Wind shear has been a primary focus. Khattak et al. [4] and Wiley [5] used wind-related variables to detect instability. Rule-based systems such as Marina et al. [16] incorporated sequences of meteorological events.
- *Machine learning classifiers:* Dhief et al. [2] and Dai et al. [10] employed tree-based and logistic models. Although interpretable, these approaches struggle to capture temporal dynamics.
- *Trajectory learning:* Liu et al. [15] developed an LSTM model for JFK airport, integrating runway configuration and in-trail spacing. Figuet et al. [14] combined GAM with ADS-B data, but achieved limited performance in fine-grained prediction.
- *Human-centered studies:* Ahlstrom et al. [3] and Dehais et al. [13] analyzed pilot scanning and workload, but their reliance on cockpit data restricts applicability to ATCO systems.

Recent advances in *Artificial Intelligence* and *machine learning* have created new opportunities for predictive modeling in aviation [2], [4], [14], [15]. Traditional go-around identification methods rely heavily on predefined thresholds and historical patterns, which limit adaptability under dynamic real-time conditions. In contrast, *deep learning* enables the construction of flexible, data-driven models that can continuously evaluate flight parameters to estimate the probability of an aborted landing.

This study addresses the following research questions:

- Can a deep learning model accurately predict the probability of a go-around before it occurs?
- How does the proposed model compare to traditional machine learning approaches?
- Which flight parameters most strongly influence go-around events?
- Is it feasible to design a universal model applicable across different airports?

We propose a deep learning-based approach to real-time go-around prediction, building on prior research conducted at the *Centro de Referencia de Investigación, Desarrollo e Innovación ATM* (CRIDA). Historical trajectory and weather data from major Spanish airports, LEBL (Barcelona-El Prat) and LEMD (Madrid-Barajas), are used to train several model architectures, including *long short-term memory* (LSTM), LSTM with attention mechanisms, and *transformer* models, in which a comprehensive set of engineered features (e.g.,

runway alignment, heading deviation, distance to runway threshold, aircraft energy, wind components) is incorporated to improve accuracy.

This work advances safer and more efficient air traffic management by delivering a real-time go-around prediction system and integrating it into a prototype interface (LandIA). The results provide critical foresight to ATCOs and represent a significant step toward intelligent aviation decision support.

II. Real-time go-around probability prediction system

A detailed overview of the workflow is presented in Fig. 1.

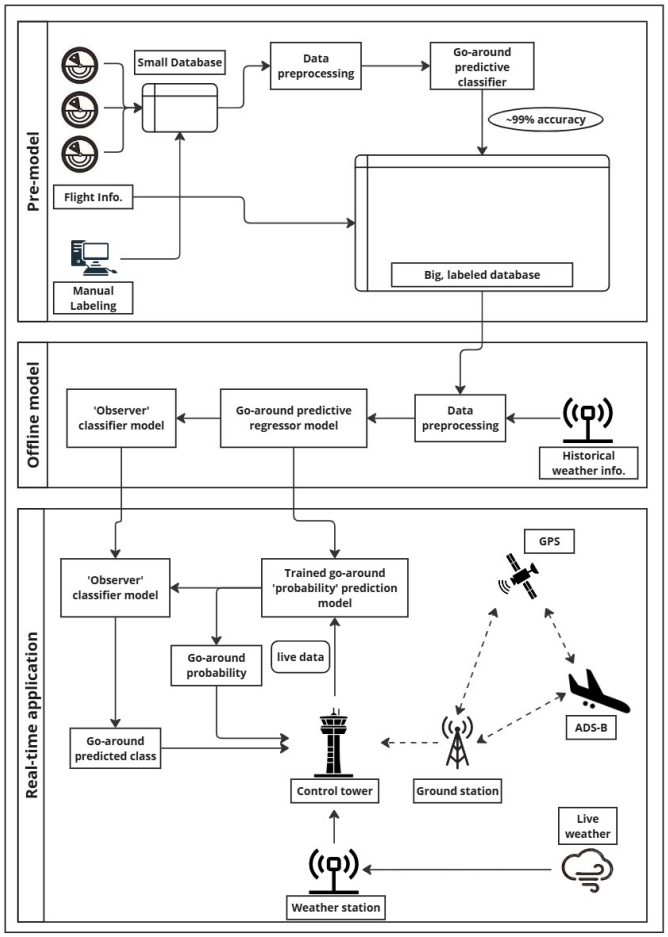


Figure 1. Workflow of the proposed real-time go-around probability prediction system.

The proposed system follows a data-driven pipeline structured into four stages: data preprocessing, feature engineering, model architecture design, and evaluation.

The project uses a pre-labeled flight database constructed via a historical XGBoost classifier [23], which distinguishes historical aborted landings from nominal operations. This is a really easy task for a machine learning (ML) algorithm, so the model obtained a precision upwards of 99%. This labeling serves as the foundation for supervised deep learning training.

This work frames go-around prediction as a *time series extrinsic regression* (TSER) task [6], where the goal is to estimate a continuous probability over time. Unlike time series classification (TSC), which maps input to discrete labels, TSER outputs a continuous likelihood — in this case, the probability of a missed approach.

Classical regression algorithms are inadequate for this task, as they often fail to capture temporal dependencies. Deep learning, by contrast, enables modeling of time-evolving flight parameters, thereby improving prediction robustness. This allows the model to both capture the temporal nature of the problem and provide real-time updated probabilities.

Once the model is successfully developed, it can be deployed in a real-time system that fetches the latest information from several sources and supplies it to the model to generate accurate predictions.

A. Data Preprocessing

The dataset comprises historical approach trajectories from Barcelona-El Prat (LEBL) and Madrid-Barajas (LEMD) airports (the airports with the highest traffic density and number of absolute GAs) between 2017–2023, enriched with meteorological data obtained from *Open-Meteo* [24]. Trajectories were extracted from radar surveillance data, with trace points recorded every 5 seconds. Outliers, incomplete tracks, mislabeled flights and alternative flights, like recreational or military, together with *light* and *jumbo* weight categories, were removed.

Each approach was truncated to the last 70 trace points before landing or go-around initiation, corresponding with the *Final Approach Fix* (FAF).

Given that go-arounds (GAs) are a rare occurrence (representing only approximately 0.2% of all approaches) some balancing strategy was imperative. Without such a strategy, the model would be heavily biased toward predicting successful landings, thus compromising its effectiveness. We employed an undersampling technique, reducing the number of successful landing instances to match the number of missed approaches (approximately 2,500 samples). We also experimented with a slightly unbalanced dataset to test whether this yielded better performance, as it might reduce conservatism in predictions and highlight differences more effectively.

Go-arounds can be triggered by a wide range of factors. Studies show that the probability of an aborted landing is strongly influenced by flight separation, approach stability, the presence of departing aircraft, and the altitude of the aircraft above the runway [10]. Among these, unstable approaches are particularly critical, as they have been identified as major contributors to commercial aviation accidents during the landing phase [11], [12].

A stable approach typically requires compliance with specific criteria regarding configuration, speed, and flight path alignment. When these criteria are not met, the approach is classified as unstable, significantly increasing the likelihood of a go-around or runway incident. Furthermore, an analysis of pilot performance and visual scanning behavior during go-arounds revealed that nearly two-thirds of pilots made errors,

often involving substantial deviations from the intended flight path [13].

Our objective is not to predict GAs triggered by unforeseeable events —such as animals or unexpected obstacles on the runway— that are not captured in the available data. Nevertheless, such events do occur and are included in our dataset. It is essential to remove these entries to ensure the proper functioning of the model.

ATCOs submit reports to a repository, *eDina*, stating the reason behind every GA. Using this information, we are able to directly omit those GAs that do not fall into any of the desired categories, especially those related to obstacles on the runway.

The database used in this study lacks instance-level labels for training. Each flight is only labeled as either an aborted or successful approach, which raises the question of how to assign labels to individual training instances within a flight trajectory.

The most straightforward method is to label each trace point during a missed approach as aborted (1), and all others as not aborted (0). However, this method can be somewhat counterintuitive when considering the dynamics leading up to a go-around. Missed approaches typically occur during the final phase of the approach, and it is often unclear—especially at earlier points—whether an abort will take place. To address this, a more nuanced labeling strategy is proposed. Previous works, such as [2], have assumed a linear increase in abort probability leading up to the go-around. While simple, this approach may be overly artificial and unrealistic.

Therefore, we experimented with a sigmoid-like probability function, in which the probability remains near zero in the early phase of the approach, gradually increases, and then sharply rises to values close to one as the aircraft initiates the go-around. This probability remains high throughout the go-around maneuver, as illustrated in Fig. 2. Both labeling approaches were evaluated, and the most appropriate one was selected based on their training performance.

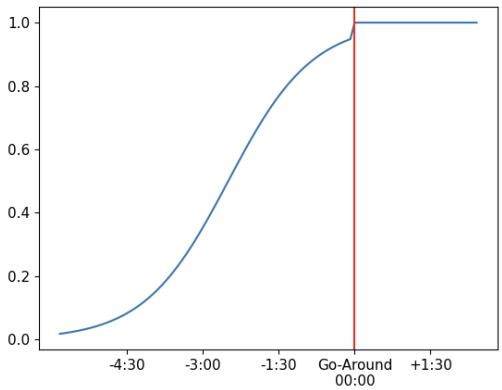


Figure 2. Expected probability evolution curve for GAs

B. Feature Engineering

We engineered time-varying features that quantify runway geometric alignment, descent and range geometry, aircraft energy state, and traffic spacing. These variables are computed at 5-second intervals throughout the final approach segment. The underlying geometry is illustrated in Fig. 3.

Figure 3. Visualization of the calculated features

- *Runway alignment*: *Heading difference* captures misalignment between aircraft and runway headings:

$$\Delta hdg = (hdg_p - hdg_A + 180) \bmod 360 - 180.$$

Let x_0, y_0 be the easting and northing differences between the aircraft position and the runway threshold, and let $\theta = (90^\circ - hdg_A) \bmod 360^\circ$ be the angle of the runway centerline. We use two complementary spatial cues. Lateral deviation ($\Delta align$) measures cross-track displacement, whereas angular deviation ($\Delta\varphi$) increases the penalty for cross-track error as the aircraft approaches the runway threshold:

$$\Delta align = \left| x \sin\theta - y \cos\theta \right|,$$

$$\Delta\varphi = \arccos\left(\frac{x\cos\theta + y\sin\theta}{\sqrt{x^2 + y^2}} \right).$$

This distinction is illustrated graphically inFig. 3.

- *Range and descent geometry.* The ground range to the runway threshold, denoted as d_{hav}, is computed using the *haversine* distance [8]. Combining d_{hav} with the altitude h yields the slant range $d = \sqrt{h^2 + d_{\text{hav}}^2}$, allowing for the calculation of the descent angle (neglecting Earth curvature):

$$\alpha = \arctan\left(\frac{h}{d_{\text{hav}}} \right).$$

- *Energy state.* Specific energy per unit weight summarizes kinetic and potential energy:

$$E_s = h + \frac{v^2}{2g},$$

with v the groundspeed and g gravitational constant.

- *Traffic spacing*. At each timestamp, the distance to the preceding arrival is computed by sequencing flights at the 4 NM fix (which is specific to each runway). If no valid predecessor is available, a large sentinel distance is assigned.

All features were normalized using standard scaling. The final set of features is presented in Table I.

C. Model Architectures

Three deep learning architectures were implemented and compared:

- *LSTM:* a baseline recurrent model capturing temporal dependencies [20].

TABLE I. FINAL SET OF FEATURES

Feature	Description
vel_z	Vertical speed
hdg	Heading of the aircraft
vel	Ground speed of the aircraft
height	Altitude above sea level
runway	Runway the aircraft is headed to
hdg_diff_rwy	Heading difference between aircraft and runway
dist_rwy	Distance to runway threshold
angle_descent	Descent angle (runway threshold as reference)
lateral_deviation	Lateral deviation of the aircraft
angular_deviation	Angular deviation (runway threshold reference)
dist_rwy_end	Distance to runway stop end
angle_final_rwy	Angle relative to the runway stop end
specific_energy	Specific energy of the aircraft
wake	Wake category of the aircraft
distToPrecedent	Distance to precedent aircraft
wakePrec	Wake category of the precedent aircraft
directionWind	Wind direction at runway level
knotsWind	Wind speed at runway level

- *Attention-based LSTM:* extends LSTM with a self-attention layer to highlight critical time steps.
- *Transformer:* a sequence-to-sequence architecture capable of parallel temporal modeling, which has proven to be state-of-the-art across multiple fields [9].

Each model outputs a probability of go-around at every second during the approach. The models are trained using binary cross-entropy as the loss function, optimized with the *Adam* algorithm, which consistently yielded stable performance. To prevent overfitting, training was regularized with early stopping based on validation set results.

D. Evaluation Protocol

The dataset was first shuffled across flights and then divided into training, validation and test subsets to simulate operational deployment. Due to the time series nature of the problem, it was not possible to use classical evaluation metrics. Although the model might output something like [0.01, 0.03, 0.08, ..., 0.97, 1.00] for a missed approach, these values do not have an absolute meaning on their own. The probability scale is entirely learned and might not align linearly with the concept of *certainty*. Our interest lies in the evolution of the signal and its ability to anticipate a missed landing.

As a first evaluation metric, we introduced a baseline measure that compares the predicted probability with the ideal binary outcome (1 for a go-around, 0 otherwise). This metric, referred to as the *Distance to Ideal* (DTI), is computed as:

$$DTI = \frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{n}\left|\hat{y}_i^{(j)} - y^{(j)}\right|, \tag{1}$$

where n is the length of the flight sequence presented to the model, N is the size of the test set, $\hat{y}_i^{(j)}$ represents the predicted probability at time step i of the j-th flight, and $y^{(j)}$ is the binary label for the j-th flight (1 if the flight results in a go-around, 0 otherwise).

Observing recurring patterns in the probability curves, we introduced a second model — an observer, so to speak. A *random forest* classifier was trained to interpret the temporal evolution of these probabilities and to decide, at a given trace point, whether the flight will ultimately execute a missed approach. This transforms a continuous probabilistic signal into a sequence of binary decisions, enabling ATCOs to receive both probabilistic and categorical information for interpretation. Additionally, this then led to us being able to quantify the quality of the classification.

1) Accuracy over time: The *random forest* observer was trained on sequences of probabilities, each associated with the true outcome (missed or successful landing). For each timestamp t, we can extract the TP, TN, FP, FN counts, establishing a *confusion matrix* at each time step. With this information, we can then calculate the accuracy (together with other metrics) with the usual formulas. This is, how well it predicts the outcome based solely on the partial probability sequence up to time t.

Plotting these metrics over time provides a clear view of how early and confidently a go-around can be anticipated. Ideally, the observer should detect a potential GA early, with accuracy growing as the aircraft approaches the missed approach point. Traditional metrics cannot capture this temporal quality, so we propose a new evaluation metric.

2) Area under the accuracy curve (AUAC): AUAC quantifies how early and consistently accurate a model is in predicting a missed approach, reflecting the intuition that earlier correct predictions are operationally more valuable.

Let acc_t denote the model's accuracy at time step t, and let T represent the total number of time steps leading up to the missed approach. The AUAC is defined as the area under the curve formed by plotting acc_t over time, for $t = 1, 2, \ldots, T$.

To compute this area, we apply the trapezoidal rule, which approximates the integral by dividing the area under the curve into trapezoids rather than rectangles:

$$\text{AUAC} = \sum_{t=1}^{T-1}\frac{(acc_t + acc_{t+1})}{2}.$$

This metric rewards not just eventual accuracy, but also how early the model becomes reliably accurate.

Its value is bounded above by T and it serves as a proxy for both early and reliable detection. In our setup, $T = 70$, corresponding to the last 70 trace points before the go-around (approximately the final six minutes of flight). A perfect model would therefore achieve AUAC $= 70$. Naturally, a higher *AUAC* does not necessarily imply a better model. For example, a model could achieve high accuracy closer to the go-around (i.e., with less advance notice), while performing poorly earlier in the approach. Thus, although AUAC provides a useful basis for approximate ranking, final model selection should consider a trade-off between overall accuracy and how early the model is able to make reliable predictions.

3) Confusion matrices and TP, TN, FP, FN rates: In addition, false positive and false negative rates can be assessed

using confusion matrices. However, these metrics are only applicable to predictions at specific time steps and do not capture the temporal evolution of the model's output. For this reason, confusion matrices are computed at selected time steps throughout the approach. To provide a more continuous view of classification performance over time, we also plot the rates of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) across all time steps. As the aircraft gets closer to the go-around point, TP and TN rates are expected to increase, while FP and FN rates should decrease.

In addition, we also calculate the *positive predictive value* (PPV) and *negative predictive value* (NPV) rates to answer the questions: 1. "Of all the cases predicted as positive, how many were actually positive?" and 2. "Of all the cases predicted as negative, how many were actually negative?".

$$NPV = \frac{TN}{TN + FN}. \tag{2}$$

The dataset is split into training, validation, and test sets using a 70/15/15 ratio. This setup allows us to evaluate model performance on unseen data in the test set, while using the validation set to train the observer model. Accordingly, the DTI score is reported on both the validation and test sets, whereas the AUAC score and confusion matrix-based rates are computed only on the test set.

To mitigate potential biases, the flight sequences are shuffled before being presented to the model during training. However, when evaluating on the test set, it is crucial to maintain the chronological order of each flight to prevent any forward-looking data leakage that could result in misleading performance metrics.

E. SHAP Analysis

Finally, SHAP values [7] were computed to assess feature importance and enhance interpretability in an operational context.

III. Results

Table II summarizes the comparative performance of the three architectures. The Transformer achieved the highest overall scores, followed by the vanilla LSTM, whereas the attention-based LSTM obtained the lowest performance.

TABLE II. Performance of tested models (test set).

Model	DTI	AUAC
LSTM	29.46	48.39
LSTM + Attention	29.60	47.04
Transformer	**30.37**	**49.97**

Regarding accuracy as a function of lead time before the event, all models deteriorate as the prediction horizon lengthens, but the Transformer consistently outperforms the alternatives across all ranges.

By further fine-tuning the model and re-balancing the dataset, we achieved a final AUAC score of 54 out of a theoretical maximum of 70. Different labeling strategies—binary, linear, and sigmoidal—were evaluated, with the sigmoidal approach producing the best results. In addition, feature importance analysis confirmed that all proposed features contribute meaningfully to the model's performance and were therefore retained.

The final model is a Transformer-based architecture comprising four encoder layers, each with two attention heads and 64 hidden units. Additional hyperparameter settings include a learning rate of 0.001, the use of the *Adam* optimizer, and *binary cross-entropy* loss criterion. To prevent overfitting, dropout regularization (rate = 0.3) was applied, randomly deactivating neurons during training.

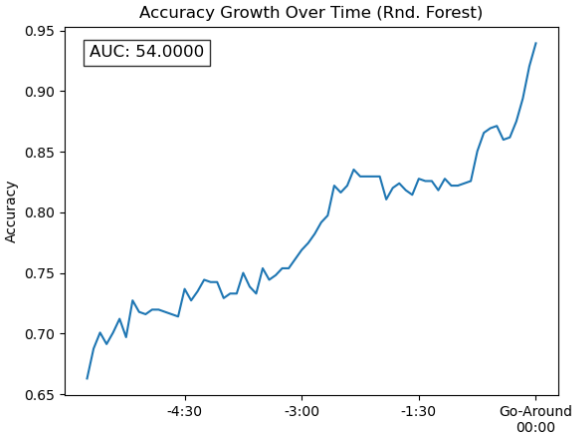


Figure 4. Prediction accuracy as a function of look-ahead time.

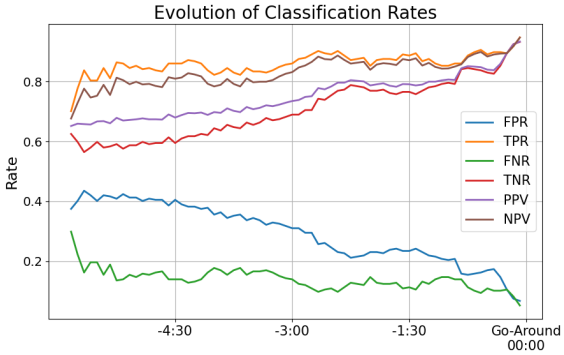


Figure 5. Temporal evolution of TP,FP,... rates as a function of look-ahead time.

Fig. 4 shows the evolution of model accuracy as a function of look-ahead time, whereas Fig. 5 and Table III report the progression of key evaluation metrics at four representative time points: 4:30, 3:00, 1:30, and 0:30, denoting the time remaining (in minutes and seconds) before either a go-around or a successful landing.

The model predicts go-arounds with 83% accuracy when forecasting 1 minute and 30 seconds in advance. As expected, prediction accuracy decreases at longer look-ahead times and improves as the aircraft approaches the go-around point, reaching 87% accuracy at 30 seconds before the event.

TABLE III. METRIC VALUES AT -4:30, -3:00, -1:30, -0:30.

Advance	*-4:30*	*-3:00*	*-1:30*	*-0:30*
Accuracy	0.73	0.77	0.83	0.87
Precision (PPV)	0.68	0.72	0.79	0.85
NPV	0.81	0.84	0.87	0.9
TPR	0.86	0.86	0.89	0.91
TNR	0.6	0.69	0.77	0.84
FPR	0.4	0.31	0.23	0.16
FNR	0.14	0.14	0.11	0.08

SHAP analysis (Fig. 6) highlights the most influential variables. Alignment-related features and in-trail spacing consistently emerge as the strongest predictors of go-arounds. Weather-related features (crosswind, headwind) rank lower, but still contribute to robustness. These insights confirm alignment with known causal mechanisms in aviation safety. Testing further demonstrated that removing less impactful features does not improve model performance.

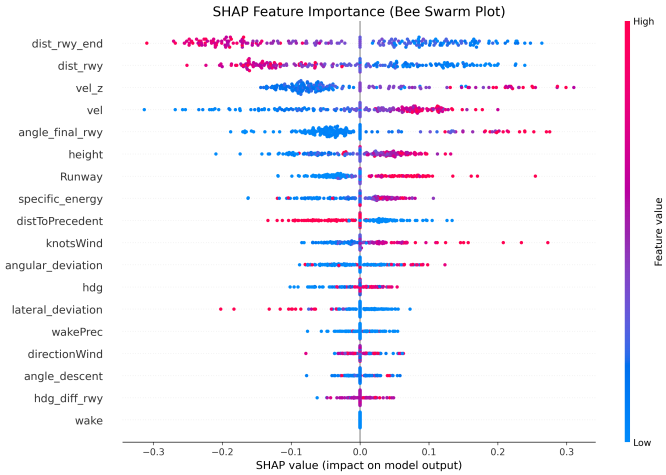


Figure 6. SHAP plot of top contributing features for transformer model.

IV. MODEL PRECISION EXPLORATION

We aim to determine whether the model is better suited to predict GAs at different stages of the approach. For this, we have decided that a missed approach will be correctly classified (the whole flight), when the observer assigns a value of 1 more frequently than 0 during the final stages of the approach. After some tests, we settled on using the last 30 trace points before the GA occurs. In other words, $30 \times 5s = 150s$, so we would be taking into account the predictions from the last two and a half minutes. Example plots are provided in an auxiliary Appendix[1], showing the probability evolution color-coded by the observer's classification (red = going to abort; blue = going to land).

By doing this, we can classify the approaches as 'predicted correctly' or not. Out of the 256 missed approaches from the test set, 209 were correctly classified by this rule and 55 were not. This represents an accuracy of 80%. With this information we can then plot the different points into a scatter plot, based on the distance to the airport at the point of abortion.

[1]The reader may find an auxiliary document in this GitHub repository.

We obtain the results in Fig. 7. It appears that non-detected

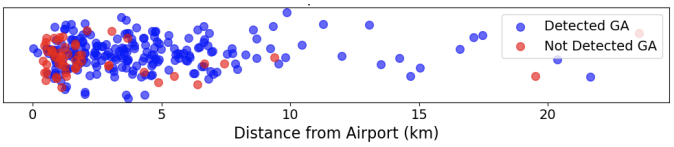


Figure 7. Distance at Go-Around initiation: detected vs non-detected

GAs were initiated closer to the airport. However, to be certain, we will check the distributions. Fig. 8 shows a histogram-like distribution of these data points, together with a *kernel density estimation* (KDE) plot [21]. Scales on the y-axes correspond to the histogram (left) and to KDE plot (right). The figure illustrates the probability distributions of the two groups, suggesting a clear distinction between cases and indicating that the model performs better at predicting go-arounds initiated farther from the airport compared to those occurring closer.

We conducted a two-sample Kolmogorov–Smirnov test to assess whether the two samples originated from the same distribution [22]. The results indicate a statistically significant difference between the distributions ($p = 2.4330e - 10 << 0.05$). In summary, we have demonstrated that the model predicts GAs more accurately when they are initiated at greater distances from the airport ($\sim > 2.5km$).

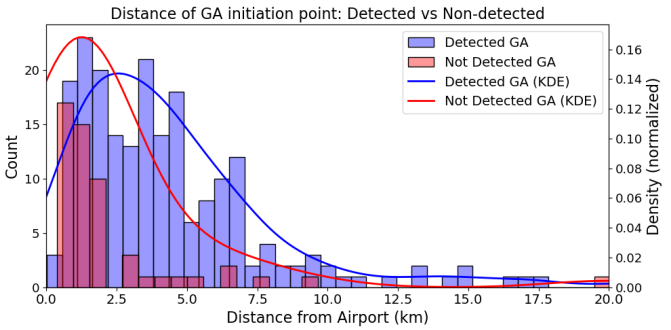


Figure 8. Histogram and KDE plots of the distributions of detected vs. non-detected GAs.

A. LEBL model in LEMD

This raises the question of how a similar model would perform in LEMD. Up to this moment, we had only tested in LEBL, so it is natural to examine how a model trained exclusively on LEMD flights would behave. Using the same settings and hyperparameters as the final model described in Section III, we obtain a well-performing model that works similarly.

However, this leads to the further question of whether a model trained with LEBL flights would be able to make accurate predictions in LEMD. Our hypothesis was that, because we calculated many features such as lateral and angular deviation from the runway and distance to its threshold, the model was sufficiently abstracted to generalize across any runway and airport.

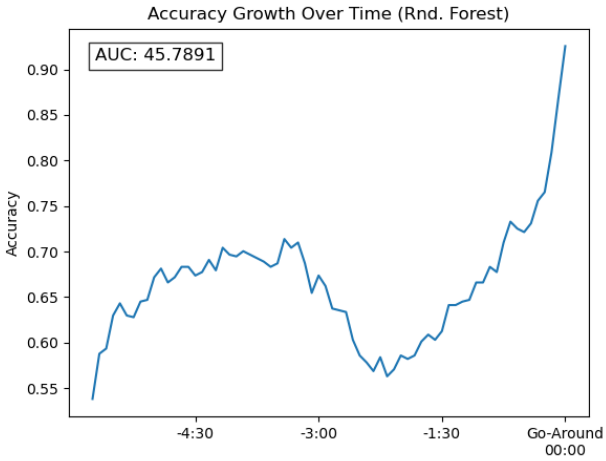


Figure 9. Accuracy curve of the LEBL model in LEMD environment

At first, the model seems to not be able to make accurate predictions. A SHAP analysis proposes the idea that certain variables related to runway distance are confusing the model, which makes sense due to the different runway dimensions across airports. Removing these variables significantly improved model interpretability and yielded acceptable predictions. Final results can be seen in Fig. 9.

In general, although the results are not as good as desired (the same model obtains a DTI of 30 and an AUAC of 49 in LEBL), it still keeps some predictive capabilities even when not having been trained on any data from LEMD.

Thus, we have demonstrated that achieving generalizability in the model requires sacrificing certain location-specific information—such as runway end limits—which may result in a slight decrease in local performance. In other words, while a fully featured model is optimal for a specific airport, a secondary model that excludes runway-specific features can still perform reasonably well across different airports.

This approach is particularly valuable for smaller or remote airports that lack sufficient data to train dedicated models. In such cases, a model trained on data from an airport with similar operational characteristics can still provide meaningful predictions of aborted landings, offering a practical alternative to the absence of predictive capability.

B. A real-time scenario

Up to this point, all experiments were conducted using historical data, which naturally makes one wonder how the model would perform in a real-time scenario. To address this, we collect live flight information from *automatic dependent surveillance–broadcast* (ADS-B) sources.

Since the entire analysis was carried out for Barcelona–El Prat Airport (LEBL), the real-time model was also deployed there. The process begins by filtering all flights within a 10-kilometer radius of LEBL. We then exclude flights that are either grounded or not conducting an approach to the airport. Most of the required information is retrieved directly from the API. However, engineered features—such as lateral deviation, distance to the runway threshold, and others—must be computed in real time for each trace point. These computations are handled efficiently and executed nearly instantly. A key challenge, though, lies in determining the runway assignment for each approaching flight. Since this information is typically not included in the ADS-B feed, we must infer it manually based on aircraft position, heading, and proximity to each runway.

For this, we take a flight current position and heading, project a point 5 NM ahead along that heading, and evaluate each runway at the airport as follows:

1) Calculating the perpendicular distance from the flight projected path to each runway's threshold.
2) Measuring the geographic distance between the aircraft position and the runway start point.
3) Ensuring the aircraft's heading is within ±25° of the runway's alignment.
4) Selecting the runway with the smallest perpendicular distance (and if tied, the shorter geographic distance).

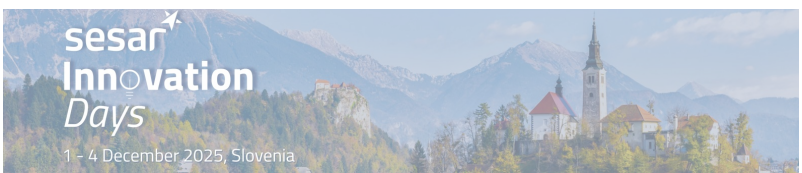
Once a target runway is assigned, we can use this information to establish a landing order on each runway. This allows us to identify the preceding aircraft for each approach and therefore compute the in-trail spacing. Additionally, we extract the wake categories of the aircraft and its predecessor and retrieve real-time wind direction and speed.

Due to the memory mechanism of the main regressor model, we load a separate instance of the trained model for each flight. That is, once an aircraft enters the 10 km monitoring range, a new model instance is initialized for that flight and receives only its updates. This means we will parallelly run two or three models, but this does not make an impact on performance.

On top of that, each model output is saved and stored individually for each flight. By doing this, we can load the observer model (Random Forest) and show it the outputs so it can interpret them and classify the flights.

For visualization purposes, we developed a web-based prototype, called *LandIA* (see Fig. 10), where we show real-time landing approaches at LEBL on a map. Flights are color-coded based on the aborted risk offered by the regressor model. On the right side, you can find a list of the current approaches, together with some other relevant information, including the outputs of both the regressor model and the observer, aircraft model, etc. The display is updated every second, which gives a good sense of continuity. If less frequent updates are acceptable, the interval can be adjusted to reduce system load; conversely, it can be increased if faster updates are required, within the limits of computing capacity.

At the time of publishing, models have been trained for both LEBL and LEMD and can be implemented into their own *LandIA* interface. Further models can be easily trained for more airports, given the data is processed correctly. As ATCOs only work in one airport at a time, it is not considered necessary to offer the visualization of more than one airport at a time; however, these visualizations could be provided with minimal work, as the code should only be slightly adapted.



The main objective of this platform is to assist ATCOs, complementing the visualizations they already use. If granted access, the system could be seamlessly integrated into existing interfaces, providing additional insights into flight status and predictive outcomes.

V. Conclusions

The results of this project show that it is possible to develop a robust model that accurately estimates the probability of an aborted landing during the final approach phase. By combining aircraft trajectory data with engineered features like heading deviation, specific energy, distance to the previous arrival and meteorological conditions, the system was able to anticipate go-arounds with considerable precision, achieving up to 87% accuracy 30 seconds before the maneuver and over 80% two minutes in advance.

Among all the evaluated architectures, transformer-based models proved to be the most effective, outperforming LSTMs in terms of both prediction stability and scalability. The model achieved a DTI score of 27.47 and an AUAC of 54, marking the best performance across all tests, further demonstrating the flexibility of transformers across a wide range of tasks.

Moreover, SHAP analysis confirmed that the model was learning the correct patterns: proximity to the runway threshold and end, vertical and absolute speed, altitude evolution, runway assignment, and alignment deviation were key contributors to go-around decisions. These findings not only validate the predictive power of the model, but also make its decisions explainable and therefore more trustworthy from an operational standpoint.

The system was further tested in real-time using ADS-B data, demonstrating that it could be integrated into a live ATCO interface, such as the *LandIA* tool developed during the project. While the model performed best when trained on airport-specific data (e.g. LEBL), it showed promising generalizability to other airports (e.g. LEMD), especially when adjusted to account for runway-dependent features.

However, several limitations remain. The current dataset uses 5-second intervals, which may smooth out rapid changes critical for decision-making. Some flight anomalies are also caused by ATC instructions or rare events not captured in the data. Despite these limitations, the model offers a substantial step forward toward real-time decision support for ATCOs.

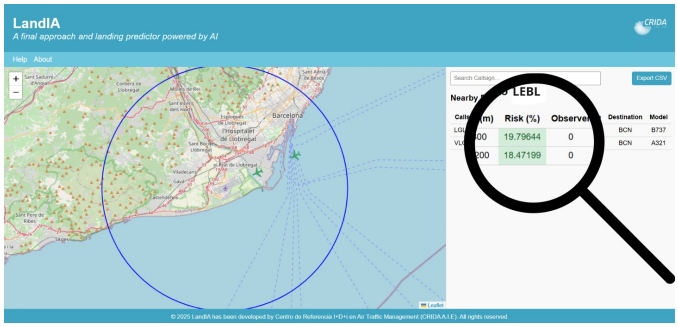


Figure 10. *LandIA* website (retrieved 23.06.2025 14:28:32)

We propose as future work improving data quality by incorporating additional meteorological information (wind shear, fog, visibility,...), data from other sources (e.g. OpenSky) and information on deceleration rates or precise pilot inputs.

Additionally, if domain experts were able to analyze individual flight trajectories and determine the point at which an aircraft begins to deviate or when a go-around becomes likely, a more refined labeling strategy could be developed—potentially enhancing model performance.

Another promising direction involves the development of interval-specific models. Instead of relying on a single model trained across the entire approach segment, one could train specialized models for specific intervals (e.g., between the 1000 ft and 800 ft altitude marks). These models may learn more fine-grained patterns and yield improved performance within their targeted range.

Finally, integrating external predictive models could further enhance our system. Previous studies have addressed related tasks, such as detecting unstable approaches [17], predicting runway occupancy times [18], and estimating runway conditions using weather and flight data [19].

Acknowledgment

This work was supported by the grants PID2021-122209OB-C31 and PID2024-155179NB-C22 funded by MICIU/AEI/10.13039/501100011033.

References

[1] EUROCONTROL and Flight Safety Foundation, *Go-Around Decision Making*. 2013.

[2] I. Dhief, S. Alam, N. Lilith, and C. C. Mean, "A machine learned go-around prediction model using pilot-in-the-loop simulations," *Transportation Research Part C: Emerging Technologies*, vol. 140, p. 103704, 2022.

[3] U. Ahlstrom and F. J. Friedman-Berg, "Using eye movement activity as a correlate of cognitive workload," *International Journal of Industrial Ergonomics*, vol. 36, no. 7, pp. 623–636, 2006.

[4] A. Khattak, J. Zhang, P.-W. Chan, F. Chen, A. Hussain, and H. Almujibah, "Wind shear and aircraft aborted landings: A deep learning perspective for prediction and analysis," *Atmosphere*, vol. 15, no. 5, p. 545, 2024.

[5] A. Khattak, P.-W. Chan, and F. Chen, "Missed approach, a safety-critical go-around procedure in low-level wind shear," *Journal of Aerospace Engineering*, vol. 2023, p. D 9119521, 2023.

[6] C. W. Tan, C. Bergmeir, F. Petitjean, and G. I. Webb, "Time series extrinsic regression," *Data Mining and Knowledge Discovery*, pp. 1–29, 2021.

[7] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *CoRR*, vol. abs/1705.07874, 2017.

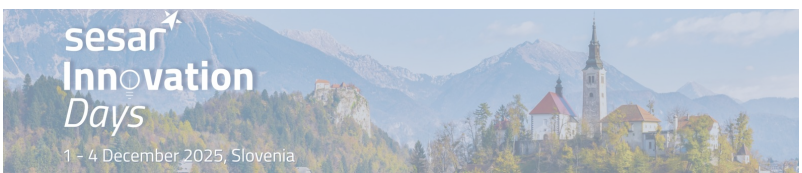
[8] R. W. Sinnott, "Virtues of the Haversine," *Sky and Telescope*, vol. 68, no. 2, p. 159, 1984.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[10] L. Dai, Y. Liu, and M. Hansen, "Modeling go-around occurrence using principal component logistic regression," *Transportation Research Part C: Emerging Technologies*, vol. 129, p. 103262, 2021.

[11] Y. Jiao, H. Sun, C. Wang, and J. Han, "Research on unstable approach detection of civil aviation aircraft," *Procedia Computer Science*, vol. 131, pp. 525–530, 2018.

[12] H.-Y. Lai, C.-H. Chen, L.-P. Khoo, and P. Zheng, "Unstable approach in aviation: Mental model disconnects between pilots and air traffic controllers and interaction conflicts," *Reliability Engineering & System Safety*, vol. 185, pp. 383–391, 2019.



[13] F. Dehais, J. Behrend, V. Peysakhovich, M. Causse, and C. D. Wickens, "Pilot flying and pilot monitoring's aircraft state awareness during go-around execution in aviation: A behavioral and eye tracking study," *The International Journal of Aerospace Psychology*, vol. 27, no. 1–2, pp. 15–28, 2017.

[14] B. Figuet, R. Monstein, M. Waltert, and S. Barry, "Predicting airplane go-arounds using machine learning and open-source data," *Proceedings of the OpenSky Symposium*, vol. 59, no. 1, p. 6, 2020.

[15] K. Liu, K. Ding, L. Dai, M. Hansen, K. Chan, and J. Schade, "Real-time go-around prediction: A case study of JFK airport," *arXiv*, 2024.

[16] C. M. Marina, J. Pérez-Aracil, E. Lorente-Ramos, C. Casanova-Mateo, and S. Salcedo-Sanz, "Go-around occurrence prediction with rule-induction, rule evolution and machine learning algorithms," *Advanced Engineering Informatics*, vol. 65, p. 103171, 2025.

[17] N. P. Singh, S. K. Goh, and S. Alam, "Real-time unstable approach detection using sparse variational Gaussian process," in *Proc. 2020 Int. Conf. Artificial Intelligence and Data Analytics for Air Transportation (AIDA-AT)*, 2020, pp. 1–10.

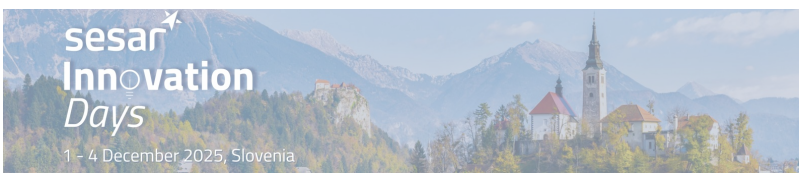
[18] H. Gao, Y. Xie, C. Yuan, X. He, and T. Niu, "Prediction of aircraft arrival runway occupancy time based on machine learning," *International Journal of Computational Intelligence Systems*, vol. 16, Sep. 2023.

[19] A. D. Midtfjord, R. De Bin, and A. B. Huseby, "A decision support system for safer airplane landings: Predicting runway conditions using XGBoost and explainable AI," *Cold Regions Science and Technology*, vol. 199, p. 103556, 2022.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] S. Weglarczyk, "Kernel density estimation and its application," *ITM Web of Conferences*, vol. 23, p. 00037, Nov. 2018.

[22] J. W. Pratt and J. D. Gibbons, "Kolmogorov-Smirnov two-sample tests," in *Concepts of Nonparametric Theory*. New York: Springer, 1981, pp. 318–344.

[23] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD '16)*, 2016, pp. 785–794.

[24] P. Zippenfenig, "Open-Meteo.com Weather API," Zenodo, 2023.