Journal of Open Aviation Science (2024), Vol.**2**
doi:10.59490/joas.2024.7943

JOAS

PROCEEDINGS | *The 12th OpenSky Symposium*

Training a Machine Learning Model to Detect Holding Patterns in Aircraft Trajectories

Xavier Olive [*,1] Luis Basora [1] Junzi Sun [2] and Enrico Spinielli [3]

[1]ONERA DTIS, Université de Toulouse, France
[2]Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands
[3]EUROCONTROL, Performance Review Unit, Brussels, Belgium
*Corresponding author: xavier.olive@onera.fr

(Received: 6 Dec 2024; Revised: 18 Mar 2025; Accepted: 17 Feb 2025; Published: 19 Mar 2025)
(Editor: Martin Strohmeier; Reviewers: Max Li and Christian Verdonk Gallego)

Abstract
This paper presents a Machine Learning (ML) model developed to detect holding pattern events in aircraft trajectories. Holding patterns are racetrack-shaped flight paths that an aircraft follows while awaiting further instructions or clearance from air traffic control (ATC). They are typically used to delay an aircraft's approach or to maintain flight without progressing towards its destination, often due to airport congestion, adverse weather conditions, or other operational factors. Accurate detection of these patterns in aircraft trajectories is crucial for performance evaluation studies within Terminal Manoeuvring Areas. Although holding patterns are relatively straightforward to define, efficiently detecting them using rule-based methods is challenging. This study details the process of labelling a dataset comprising over 130,000 aircraft trajectories landing at five major European airports and training a model to accurately identify these patterns.

Keywords: air traffic management, ADS-B, trajectory analysis, data preprocessing, machine learning

1. Introduction

As air traffic control (ATC) is responsible for ensuring safe and efficient operations, this task becomes particularly complex within Terminal Manoeuvring Areas (TMAs), where numerous aircraft converge towards one or more runways. In these areas, aircraft must reduce speed and altitude, align for landing, and maintain safe separation distances, including wake turbulence separations. The challenge intensifies under adverse conditions, such as fog or thunderstorms, which necessitate greater separation distances, further complicating the management of air traffic flow.

Control strategies within Terminal Manoeuvring Areas (TMAs) include level-offs, path stretching, point-merge techniques, and holding patterns [1]. Previous research [2] has demonstrated that holding patterns have the most significant adverse environmental impact among these strategies, regardless of the underlying cause. Another study [3] investigated the factors contributing to holding patterns at major European airports. These analyses were all based on the original detection method implemented in the traffic library [4], a method which, until now, has not been formally published in an academic context.

In this paper, we present the method used to label an original dataset [5] with holding pattern in-

formation. Initially, we applied information extraction techniques based on autoencoding neural networks, to explore the characteristics of holding patterns within the generated latent space. This approach allowed us to identify areas where holding patterns tended to cluster, resulting in a prelabelled dataset. This dataset was then meticulously and laboriously relabelled by the authors. Following this, the initial layers of the autoencoder were retained, and the downstream layers were replaced with a conventional classifier trained to specialize in the identification of holding patterns.

This approach led to the development of a highly effective model for labelling holding pattern situations, which has since been published and is now widely adopted by the community through the traffic library. Such a method is particularly helpful in monitoring airborne delays in terminal airspace, supporting the SES Performance Scheme's KPIs, including the ASMA Time metric for airborne holding. By detecting delays such as those caused by racetrack holding or vectoring using open data, it could complement ANSP data and enhance the evaluation of operational efficiency and environmental impact.

In the following, Section 2 provides the context and formal definition of a holding pattern, along with the rationale for choosing a Machine Learning approach over a rule-based detection method. The process of constructing a labelled trajectory dataset is detailed in Section 3, while Section 4 describes the training procedure. Finally, the results and their implications are presented and discussed in Section 5.

2. Definition of a holding pattern

A holding pattern is a manoeuvre where an aircraft flies a racetrack-shaped pattern in a designated area. Such a manoeuvre can be implemented en route, when the crew needs to run through checklists [6] and troubleshoot problems, or by refuelling aircraft [7]. They are often implemented in TMA as a last resort to sequence aircraft using limited space. When operations are disrupted, it is a common practice to stack holding patterns with aircraft flying the racetrack shape at various altitudes, the lower aircraft having the higher priority. We have observed that this practice varies across airports: some use holding patterns as a last resort during degraded conditions, while others implement them during periods of congestion. For instance, in London Heathrow, holding patterns are not a sign of degraded conditions, whereas they would occur in Paris area only in exceptional conditions, such as limited visibility (fog).

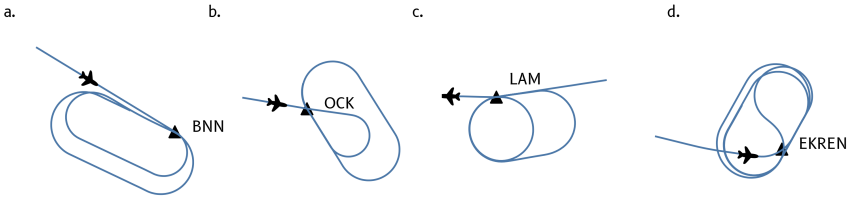

Figure 1. Holding patterns may be entered according to different patterns: direct entry (a.), tear-drop entry (b.), and some variants may also be implemented, with some oval shapes becoming circles (c.) or switching from a left-hand turn to a right-hand turn upon entry (d.)

Holding patterns are defined from a navigational point, called *holding fix* which forms the end of an inbound leg. Depending on the initial bearing of the trajectory, aircraft enter a holding with different patterns (Figure 1). Holding patterns are mostly flown in a standard direction (right-hand turns) but non-standard patterns are also common (Figure 1.c).

Historically, the racetrack shape has been preferred over circles, as the latter limit situational awareness. The introduction of RNAV made it easier to fly any pattern, but since the rules of aviation were

standardized before GPS came into common use, racetrack patterns developed for holding at the time have remained the norm.

Despite being carefully designed, holding patterns are very hard to properly label systematically due to large variances in the way to enter a holding pattern and in the duration of the straight legs, if any. Attempts to detect circles, or intervals where the track angle covers a range of 360 degrees, fail in many corner cases.

Figure 2 shows various situations for holding patterns implemented on trajectories in a terminal manoeuvring area: holding patterns have been highlighted in orange by the model we present in this contribution. An ML model allows to detect situations even when they do not perfectly match simple necessary conditions to define a holding pattern:

1. the trajectory is not self-intersecting;
2. the trajectory path is further stretched at the exit of the holding pattern;
3. two holding patterns are implemented in sequence; the first one (to the East) looks atypical;
4. the aircraft enters and exits a holding pattern without running a full loop;
5. the trajectory shows a tear-drop entry but exits the holding pattern before running a full loop;
6. this atypical trajectory with many landing attempts at Lelystad airport (EHLE) shows only one short holding pattern; other "loops" should not be labelled as so.

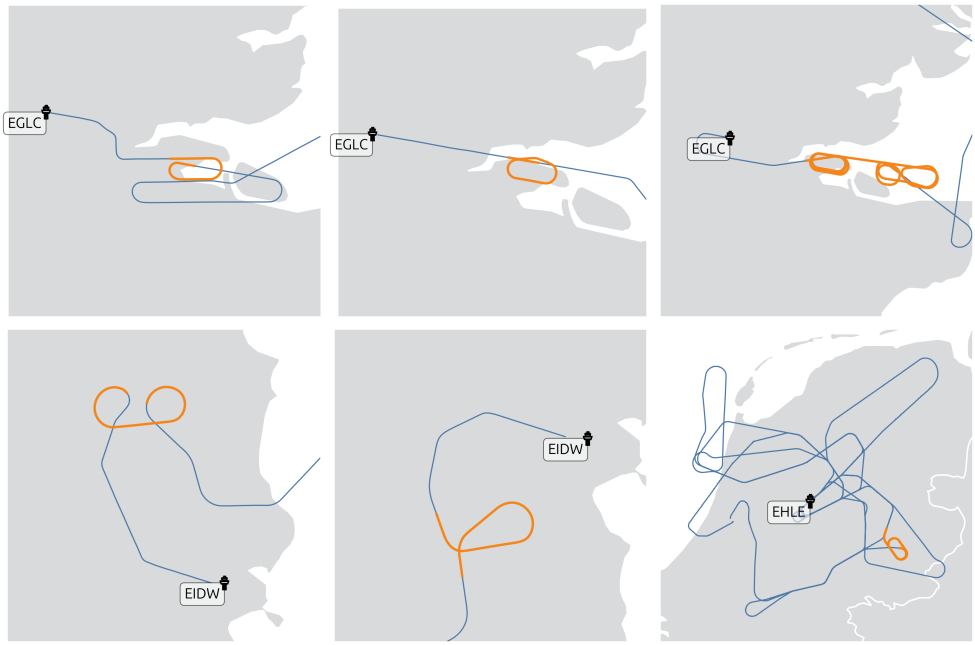

Figure 2. The proposed model effectively detects and labels various types of holding patterns. ML helps to replace fuzzy definitions, accounting for numerous corner cases, with example-driven learning for a more robust detection.

In this contribution, we prefer an ML approach over traditional rule-based methods when developing a method to detect and label various types of holding patterns. Rule based methods would rely on fuzzy definitions, which will be prone to failure when faced with corner cases or unexpected variations in the data. Machine Learning, on the other hand, allows us to train the model using real examples of holding patterns, enabling it to learn from the inherent variability in the data. By focusing on data-driven learning, we eliminate the need for a formal definition and substitute it with a substantial dataset of examples.

The drawback of this approach is that, at this time, there is no such labelled collection of examples. We explain in Section 3 how we constitute such a dataset.

3. Constitution of a labelled dataset

To build an accurate ML model for detecting holding patterns, we must first construct a properly labelled dataset. In the following, we detail the initial steps involved in creating this dataset, including how we automatically generated an initial labelling of the trajectories, which was later manually verified by the authors of this contribution.

The data used for this study is collected by the OpenSky Network [8], a network of ADS-B receivers, which offers querying capabilities on their database for academics. Recorded data contains timestamps (added on the receiver side, with many receivers equipped with a GPS nanosecond precision clock), transponder unique 24-bit identifiers (`icao24`), space-filled 8-character callsign, latitude, longitude, barometric altitude, geometric altitude, ground speeds, true track angle, and vertical speed.

Trajectories of aircraft landing at major European airports are provided (see Table 1). Trajectories are resampled and clipped to fit in a radius of a size that is different according to airports in order to capture the area where holding patterns tend to be implemented. Illustrations in this paper may use trajectories of aircraft landing at other airports, such as Zurich (`LSZH`) or Amsterdam Schiphol (`EHAM`), but those are not part of the final labelled dataset.

Table 1. Description of the datasets used in the study

airport	code	area of interest (radius)	size of the dataset	number of holding patterns
London Heathrow	`EGLL`	50 nm	38,550 trajectories	13,680 (36 %)
London City	`EGLC`	60 nm	4,364 trajectories	50 (1.2 %)
Dublin	`EIDW`	50 nm	17,457 trajectories	4438 (4.5 %)
Paris Charles de Gaulle	`LFPG`	90 nm	37,085 trajectories	78 (2.1 %)

The model's objective is to identify segments of trajectories that can be labelled as holding patterns, representing a *detection task*, as opposed to methods that simply determine whether a trajectory contains a holding pattern or not, which would be a *classification task*. To simplify this detection task, we frame it as a classification problem applied to segments of trajectories. Instead of analysing full-length trajectories directly, we divide them into overlapping segments using a sliding window approach (Figure 3).

A straightforward approach to classify data in unsupervised ML involves clustering techniques. However, traditional clustering methods face challenges when applied to trajectory data, primarily due to difficulties in defining meaningful distance metrics. A common practice is to sample the trajectory and represent it as an n-dimensional vector of points, enabling the use of point-based clustering algorithms and metrics like the Euclidean distance. Unfortunately, this approach is hindered by the curse of dimensionality. Alternative distance measures have been developed to better account for the geometry and shape of trajectories [9]. Among these, the Hausdorff distance [10] and the Fréchet distance [11] are particularly well-known.

To overcome the limitations of traditional clustering methods, we can utilize deep clustering techniques [12], which involve projecting samples into a lower-dimensional latent space and performing clustering within this reduced space. In this study, we applied a trajectory clustering technique previously introduced in [13], leveraging autoencoders to construct the latent space. Autoencoders

Figure 3. Trajectories are divided into overlapping segments using sliding windows of 6 minutes with a 2-minute shift. (For this map, slightly different values are used, and a lateral offset is applied to the segments for improved legibility.)

are particularly suited for this task, as they compress input data into a compact latent representation while preserving its essential features. Autoencoders are a powerful tool for mapping high-dimensional data into a lower-dimensional space, while effectively grouping samples with similar features together. Figure 4 visualizes the latent space generated by the autoencoder, showcasing clusters of holding pattern segments.

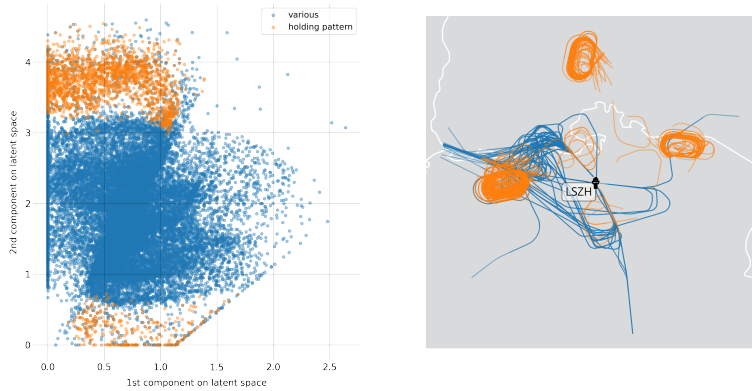

Figure 4. Latent space representation of a selection of trajectory segments, with holding patterns forming distinct clusters

Figure 5 illustrates how an entire trajectory, represented as a sequence of 6-minute segments, can be mapped onto the previously defined latent space. In this visualization, all segments (depicted as 2-dimensional points in the latent space) that fall within the orange region are identified and should be labelled as holding patterns.

For our approach, we implemented a basic Gaussian Mixture Model (GMM) to detect clusters containing holding patterns; GMM works by modelling the data as a mixture of multiple Gaussian distributions, each representing a cluster. Figure 6 shows an effective clustering achieved by the GMM with 4 components. To refine the clustering, we trained the autoencoder on a subset of trajectories

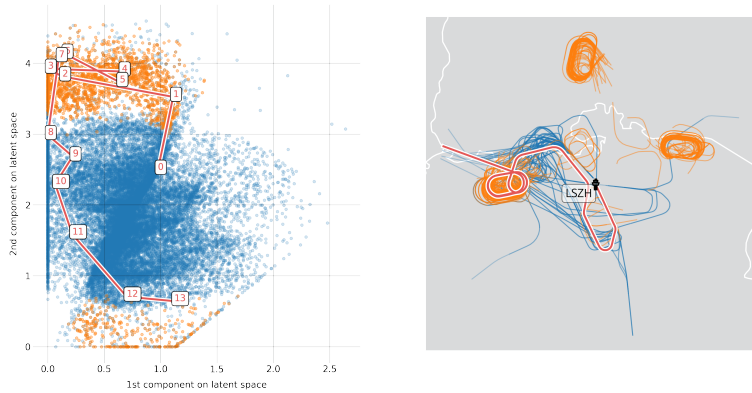

Figure 5. Example of a trajectory mapped into the latent space, together with a subset of the input trajectories

containing only those with self-intersections, which reduced the density of negative samples in the latent space and encouraged the formation of dense clusters for holding patterns.

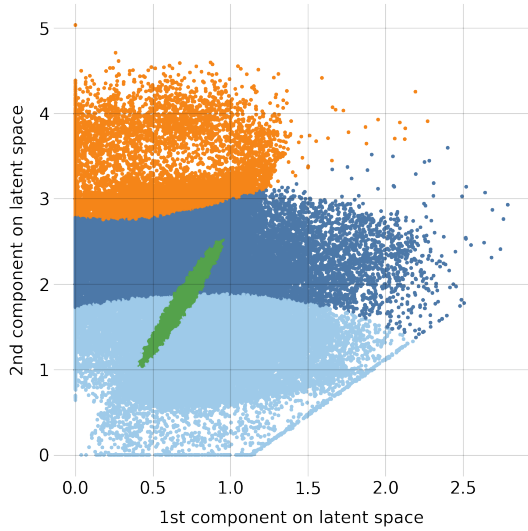

Figure 6. Resulting clustering on the latent space with a 4-component Gaussian Mixture Model approach: the orange cluster seems to capture a lot of the holding pattern segments.

The initial labelling obtained through clustering was applied to the entire dataset of trajectories, creating a pre-labelled dataset. At this stage, the performance of the initial project-then-cluster step was not critical, as the entire dataset was subsequently manually reviewed by the authors. During this exhaustive process, all false positives and false negatives were corrected to produce the final labelled dataset. This was the most time-consuming and least rewarding part of the work, yet it was crucial for the accuracy of the training part.

It should be noted that the labelling was conducted by multiple authors, each bringing their own definition on what constitutes a holding pattern. Moreover, their interpretations of holding patterns may have evolved throughout the labelling process. While this variability might be viewed as a limitation, it can also be considered a strength as it introduces variance and regularization into the dataset, all that can be beneficial during the training phase of the model (Section 4).

Technical implementation. Each trajectory was divided into overlapping sliding windows of 6 minutes with a 2-minute shift. These segments were then resampled into 30 evenly spaced points, corresponding to one data point every 12 seconds. To handle discontinuities in the track angle, the values were unwrapped to prevent abrupt jumps (e.g., from 359° to 1°) by continuing the sequence beyond 360° (e.g., to 361°). Additionally, the track angle values were normalized by shifting them so that the first timestamp starts at zero, followed by a min-max scaling (scikit-learn implementation). The processed data was projected into a latent space using a simple autoencoder with four layers. The architecture consisted of an input layer with 30 neurons, a second layer with 8 neurons, a bottleneck layer with 2 neurons, and a symmetric decoder with 8 and 30 neurons, respectively. The projection operator utilized only the first two layers, which produced the low-dimensional latent representation of the trajectory segments. The code for processing trajectories and implementing the methodology described in this section is available on GitHub and is based on the `traffic` library.

4. A supervised model for holding pattern detection

Once the dataset was fully constituted, we employed a cross-airport validation strategy and divided the dataset into training and testing subsets: models were trained on data from a subset of airports and tested on the remaining ones. As for metrics, due to the imbalanced nature of the dataset, we let accuracy aside and focused instead on precision, recall, F1-score, and Intersection over Union (IoU). Precision, recall and F1-score are implemented at the segment level (*"Is the six-minute segment part of a holding pattern?"*), while IoU is implemented at the full trajectory level. The IoU score was anticipated to be lower, given the inherent ambiguity in precisely defining the starting and ending points of a holding pattern.

We tested two architectures:

- a fully connected (FC) network resembling the autoencoder from Section 3, and
- a convolutional (CNN) network, as illustrated in Figure 7.

We trained the model on the resampled unwrapped track angle values, and compared the results with the effect of including vertical rate values into the model (which would slightly change some sizes in Figure 7).

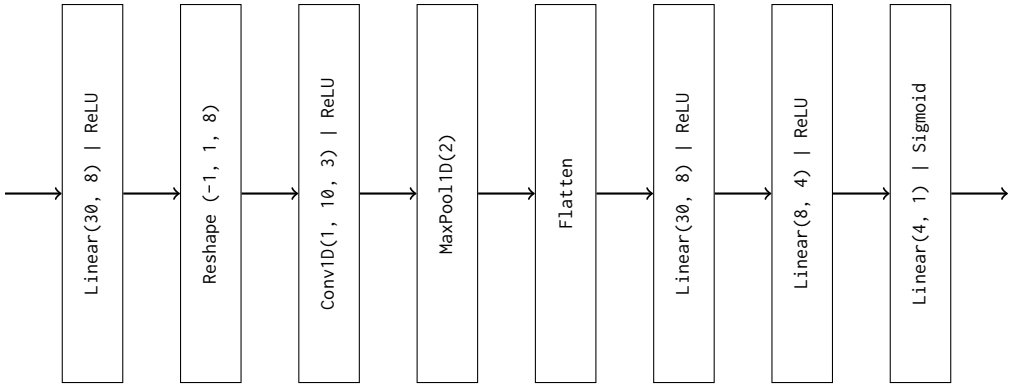

Figure 7. Structure of the convolutional architecture used for the model

A series of experiments were conducted to evaluate the performance of these architectures under various configurations, including training on subsets of airports and testing on unseen airports. The results, summarized in Table 2, indicate that the convolutional architecture generally outperformed the fully connected network in terms of precision, recall and F1-score. The highest scores were

achieved with the convolutional architecture when trained on data from EGLL, EGLC, and LFPG and tested on EIDW, using both track angle and vertical rate as input features. We also noted a substantial variability in the results was observed depending on the airport pairs used for training and testing.

Table 2. Performance Metrics

precision	recall	F1	train	test	features	architecture
0.8504	0.7005	**0.7682**	*	EIDW	track + vertical rate	CNN
0.8358	0.6959	0.7594	*	EIDW	track	CNN
0.7628	**0.7542**	0.7584	EIDW	*	track	CNN
0.7428	0.7704	0.7563	EIDW	*	track + vertical rate	CNN
0.7891	0.7009	0.7423	EGLL	*	track	CNN
0.7752	0.6619	0.714	EGLL	*	track + vertical rate	CNN
0.395	0.6267	0.4845	*	EGLC	track	FC
0.3655	0.7067	0.4818	*	EGLC	track + vertical rate	CNN
0.3472	0.6757	0.4587	*	LFPG	track	CNN
0.3063	0.6622	0.4188	*	LFPG	track + vertical rate	CNN

Including the vertical rate provided a marginal improvement in performance across most configurations, which led us to publish the second model in the list, using only track angle values, trained on London and Paris airports and tested on Dublin airport.

The model was also tested on less typical data, such as practice go-arounds and aerial surveys, yielding successful results. Although the dataset of these atypical trajectories is included in the `traffic` library's set of sample trajectories, it is not large enough to perform meaningful statistical analysis.

5. Discussion and conclusion

In this contribution, we present the approach adopted to develop a model capable of detecting holding patterns in aircraft trajectories. While the model was trained and tested on labelled data from four airports, it has demonstrated strong generalization capabilities, effectively labelling trajectories from different contexts, as shown in Figure 2.

The model has already been widely implemented as part of the `traffic` library for various visualizations (e.g., Figure 8) and other contributions, such as [2, 3]. Further validation has been conducted through its application to in-flight emergencies analyzed in [6], where holding patterns extend beyond terminal manoeuvring areas. The model has not shown any significant misclassification of other trajectory loops that cannot be categorized as holding patterns.

The performance of the model has been deemed satisfactory by both the authors and the broader community. However, as with many machine learning-based models, it lacks clear explainability regarding why a particular trajectory is labelled as a holding pattern or not. To assist the community in any effort to come up with a better model, the authors provide both the trajectories and the corresponding labels alongside this contribution.

Author contributions

Conceptualization (X.O), Methodology (*all*), Software (X.O, L.B), Validation (*all*), Formal analysis (*all*), Investigation (X.O, L.B), Data Curation (X.O, L.B), Writing – Original Draft (X.O), Writing – Review & Editing (*all*), Visualization (X.O), Project administration (X.O), Funding acquisition (X.O)

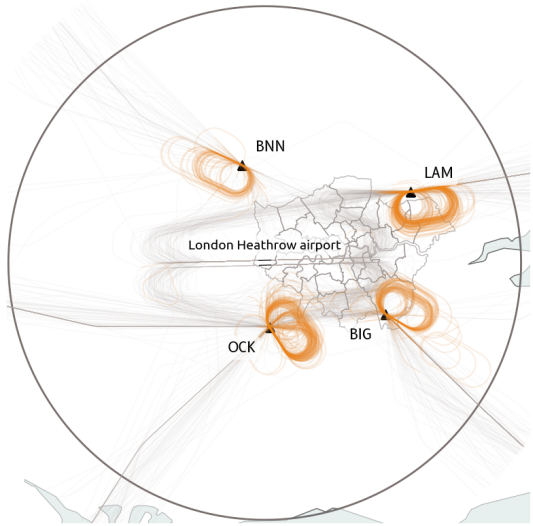

Figure 8. Holding patterns labelled for trajectories landing at London Heathrow Airport

Funding statement

The authors are grateful to the EC for supporting the present work, performed within the NEEDED project, funded by the European Union's Horizon Europe research and innovation programme under grant agreement no. 101095754 (NEEDED). This publication solely reflects the authors' view and neither the European Union, nor the funding Agency can be held responsible for the information it contains.

Open data statement

The resulting data has been made available on the 4TU.ResearchData repository [5] and are available as imports from the `traffic` library.

The models are delivered as onnx files in the `traffic` libraries. The models are subject to the MIT licence, like the rest of the library. They can be freely reused in other software, regardless of the programming language. They must however remain credited.

Reproducibility statement

The Python scripts used to build the dataset and train the model are available on a GitHub repository: https://github.com/xoolive/holding_patterns.

References

[1] Henrik Hardell, Anastasia Lemetti, Tatiana Polishchuk, and Lucie Smetanová. "Evaluation of the Sequencing and Merging Procedures at Three European Airports Using Opensky Data". In: *Proceedings of the 9th OpenSky Symposium*. Brussels, Belgium, Nov. 2021. DOI: 10.3390/engproc2021013013.

[2] Xavier Olive, Junzi Sun, Luis Basora, and Enrico Spinielli. "Environmental Inefficiencies for Arrival Flights at European Airports". In: *PLoS ONE* 18.6 (June 2023). DOI: 10.1371/journal.pone.0287612.

[3] Ramon Dalmau, Philippe Very, and Gabriel Jarry. "On the Causes and Environmental Impact of Airborne Holdings at Major European Airports". In: *Journal of Open Aviation Science* 1.2 (Oct. 2023). DOI: 10.59490/joas.2023.7198.

[4] Xavier Olive. "traffic, a toolbox for processing and analysing air traffic data". In: *Journal of Open Source Software* 4.39 (2019), pp. 1518–1. DOI: 10.21105/joss.01518.

[5] Xavier Olive, Luis Basora, and Junzi Sun. *Arrival Trajectories at Five Major European Airports*. Aug. 2022. DOI: 10.4121/20411868.V1.

[6] Xavier Olive, Axel Tanner, Martin Strohmeier, Matthias Schafer, Metin Feridun, Allan Tart, Ivan Martinovic, and Vincent Lenders. "OpenSky Report 2020: Analysing in-flight emergencies using big data". en. In: *Proceedings of the 39th IEEE/AIAA Digital Avionics Systems Conference (DASC)*. 2020, p. 10.

[7] Xavier Olive, Junzi Sun, Adrien Lafage, and Luis Basora. "Detecting Events in Aircraft Trajectories: Rule-Based and Data-Driven Approaches". en. In: *Proceedings of the 8th OpenSky Symposium*. Dec. 2020. DOI: 10.3390/proceedings2020059008.

[8] Matthias Schäfer, Martin Strohmeier, Vincent Lenders, Ivan Martinovic, and Matthias Wilhelm. "Bringing up OpenSky: A large-scale ADS-B sensor network for research". In: *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*. IEEE. 2014, pp. 83–94.

[9] Philippe Besse, Brendan Guillouet, Jean-Michel Loubes, and Royer François. "Review and perspective for distance based trajectory clustering". In: *arXiv preprint arXiv:1508.04904* (2015).

[10] Felix Hausdorff. *Grundzuge der Mengenlehre*. Vol. 61. American Mathematical Society, 1978.

[11] Maurice Fréchet. "Sur quelques points du calcul fonctionnel". In: *Rendiconti del Circolo Matematico di Palermo (1884-1940)* 22.1 (1906), pp. 1–72.

[12] Sheng Zhou, Hongjia Xu, Zhuonan Zheng, Jiawei Chen, Zhao Li, Jiajun Bu, Jia Wu, Xin Wang, Wenwu Zhu, and Martin Ester. "A Comprehensive Survey on Deep Clustering: Taxonomy, Challenges, and Future Directions". In: *ACM Computing Surveys* 57.3 (Mar. 2025), pp. 1–38. DOI: 10.1145/3689036.

[13] Xavier Olive, Luis Basora, Benoıt Viry, and Richard Alligier. "Deep Trajectory Clustering with Autoencoders". In: *Proceedings of the 9th International Conference on Research in Air Transportation*. 2020.