traffic series #2

Training a Machine Learning Model to Detect Holding Patterns from ADS-B data

Xavier Olive Luis Basora Junzi Sun Enrico Spinielli

OpenSky Symposium 2024

ONERA TU Delft EUROCONTROL

Introduction

Focus on a specific method in the `traffic` library:

```
>>> flight.holding_pattern()
>>> flight.next("holding_pattern")
>>> flight.label("holding_pattern")
```

Contributions:

- a labelled dataset: doi:10.4121/20411868
  (2 months of data at `EGLC`, `EGLL`, `EHAM`, `EIDW`, `LFPG`, `LSZH`)
- a ML model (`.onnx` files: *scaler* and *classifier*)
- an implementation in `traffic`

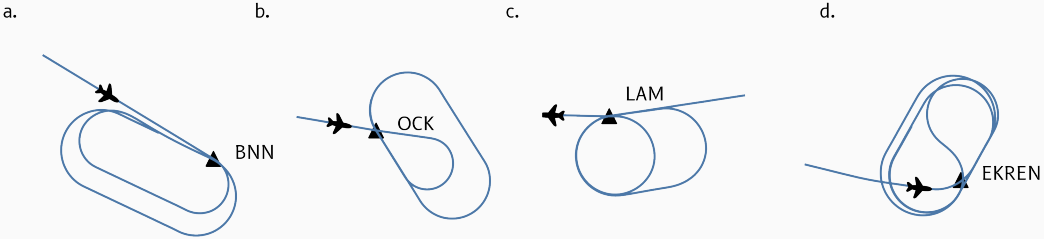
Definition

holding pattern *(noun)*: flight manoeuvre used to keep an aircraft while awaiting further instructions. Holding patterns are typically used when air traffic congestion or adverse weather conditions prevent an aircraft from landing immediately or progressing along its intended route.

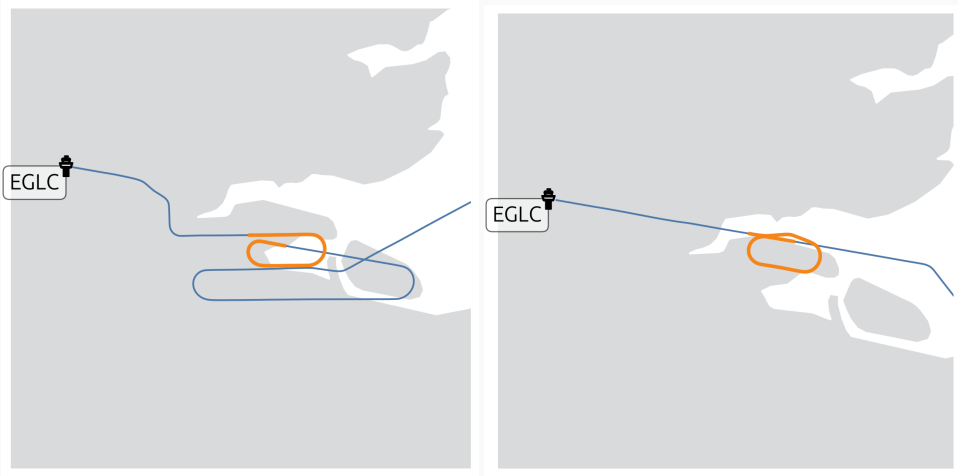
An aircraft usually follows a racetrack-like loop, typically flying a **straight path** for a set distance, then making a **turn to complete the loop**.

Rule-based method? ML-based method?

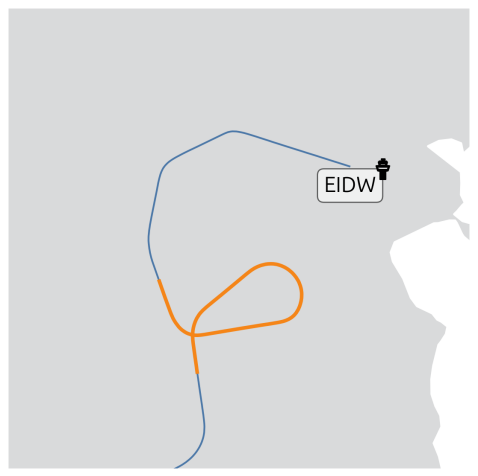
Illustration



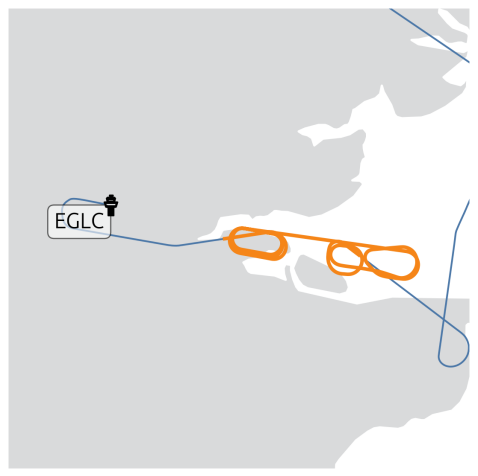
Bestiary (page 1)

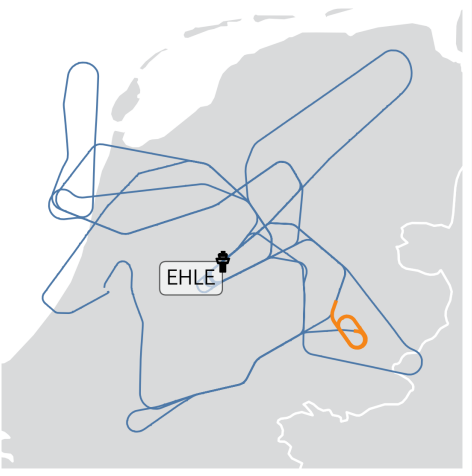


Bestiary (page 2)



Bestiary (page 3)





Approach

- We do not have a labelled dataset
- We want more than a yes/no answer: classification ➤ detection
- We know few things about *deep clustering:* build a latent space so that two resembling trajectories will be neighbours on a low dimension space

Step 1: Label a dataset (*unsupervised* learning, then manual labour)

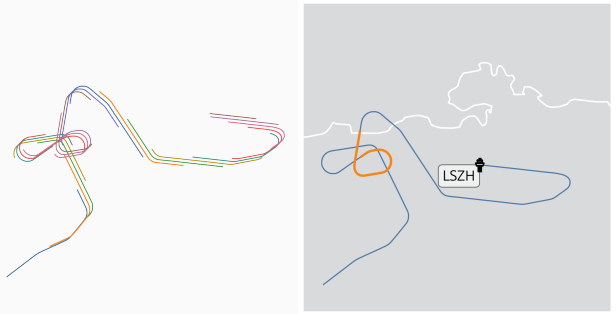
Step 2: Train a model (*supervised* learning)

Step 1: Label a dataset

Sliding windows

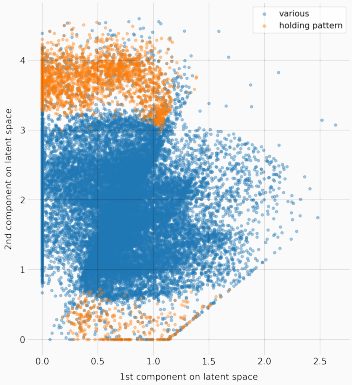
We need to identify parts of trajectories, so we will proceed with *sliding windows:*
(classification vs. detection)

```
flight.sliding_windows("6 min", "2 min")
```



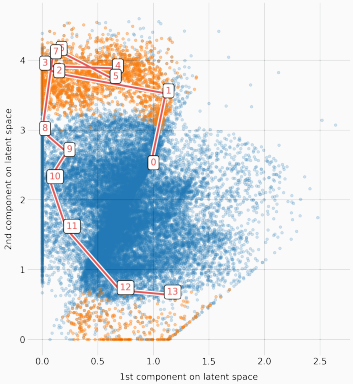
Structuration of the latent space

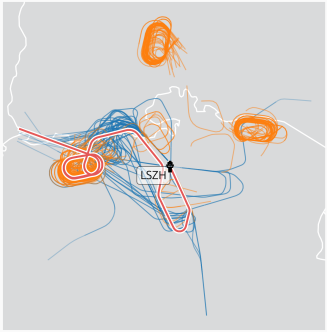
After training a basic autoencoder `30-8-2-8-30` to create a latent space, we note that holding patterns tend to cluster:



Trajectory in the latent space

We can identify the holding pattern part of the trajectory in the latent space:

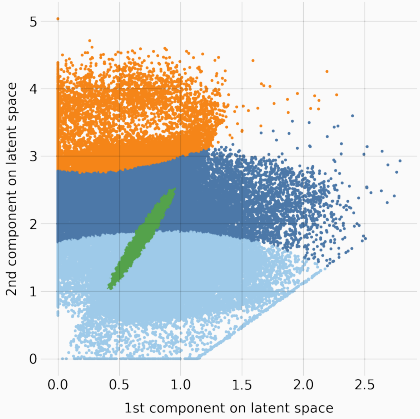




Clustering of the latent space

Now, we need to identify those “holding pattern” areas for a pre-labelling:

clustering the latent space (Gaussian Mixture for instance)



From a pre-labelling to a labelling

- Now we can mark all segments in the proper cluster as *“holding pattern”*
- But in fact, it was quite average (imagine *Google Translate level*)

We adjusted all the labels manually with a visual tool:

- left plot (*“holding patterns”*):
  click to mark as “**not** holding pattern”
- right plot (“**not** holding pattern”):
  click to mask as *“holding pattern”*

GET THE SHIT

☑ DONE

97,700 flights 784,177 segments (49,822 positive)

GET THE SHIT

☑ DONE

97,700 flights 784,177 segments (49,822 positive)

The biggest, most horrendous, terribly monstrous part of the work—the absolute pinnacle of frustration and despair, the unparalleled abyss of toil and tedium, the most impossibly dreadful and gruelling segment of it all.

Step 2: Train a classifier

Supervised learning with a NN classifier

- `flight.sliding_window("6 min", "2 min") if segment.duration >= "5 min"`
- Feature: *unwrapped track angle* | `MinMaxScaling()`
- Convolutional architecture (significantly better than flattened):

```
   Linear(30, 8) | Reshape(-1, 1, 8) | Conv1d(1, 10, 3) | ReLU()
 | MaxPool1d(2) | Flatten() | Linear(30, 8) | ReLU()
 | Linear(8, 4) | ReLU() | Linear(4, 1) | Sigmoid()`
```

- train/test splitting (~ 80%)
- Metrics: accuracy, precision, recall, f1-score, IoU

  Best results with `EIDW` *as testing set (precision: 0.85, recall: 0.75, f1: 0.76, IoU: ~0.6 but...)*

Conclusion

Key take-aways

- Unsupervised learning helps to bootstrap a labelling
- Now, we have a labelled dataset for holding patterns (cost: *blood and tears*)
- Models exported in `.onnx` format: usable in any programming language

```
from traffic.data.datasets import landing_heathrow_2019


subset = landing_heathrow_2019.has('holding_pattern').eval().sample(600)
subset.plot(color="#bab0ac")  # gray
for hp in subset.holding_pattern():
    hp.plot(color="#f58518")  # orange
```

Visuals