Deep Trajectory Clustering with Autoencoders

Xavier Olive*, Luis Basora*, Benoît Viry*, Richard Alligier†

*ONERA – DTIS
Université de Toulouse
Toulouse, France

†École Nationale de l'Aviation Civile
Université de Toulouse
Toulouse, France

***Abstract*—Identification and characterisation of air traffic flows is an important research topic with many applications areas including decision-making support tools, airspace design or traffic flow management. Trajectory clustering is an unsupervised data-driven approach used to automatically identify air traffic flows in trajectory data. Long-established trajectory clustering techniques applied for this purpose are mostly based on classical algorithms like DBSCAN, which depend on distance functions capturing only local relations in the data space.**

Recent advances in Deep Learning have shown the potential of using deep clustering techniques to discover hidden and more complex relations that often lie in low-dimensional latent spaces. The goal of this paper is to explore the application of deep trajectory clustering based on autoencoders to the problem of flow identification. Thus, we present two clustering techniques (artefact and DCEC) and show how they embed trajectories into the latent spaces in order to facilitate the clustering task.

***Keywords*— machine learning; deep learning; trajectory clustering; deep clustering; autoencoders**

I. INTRODUCTION

Identification and characterisation of air traffic flows is important for decision-making support in areas such airspace design or traffic flow management. The identification of air traffic flows from trajectories can be done automatically from trajectory data by applying trajectory clustering techniques in order to find the set of clusters which best fit the operational flows within an airspace.

Clustering is a fundamental unsupervised method for knowledge discovery. Because of its importance in exploratory data analysis, clustering is still an active field of research. The application of clustering to trajectories is particularly challenging for several reasons. First, trajectories have a functional nature and the definition of appropriate distance functions between trajectories is difficult. Secondly, trajectories are often available as samples of data points in high-dimensional space where classical distances loose their discriminative power (curse of dimensionality). Thirdly, the availability of massive amounts of open trajectory data to be explored require highly efficient and scalable trajectory clustering algorithms.

Trajectory clustering approaches relying on classical methods such k-means or DBSCAN suffer from the same issues. These well-established algorithms depend on similarity (or distance) functions usually defined directly on high-dimensional data space (trajectories), which prevent them from capturing richer dependencies potentially lying in the low-dimensional latent space. For this reason, it is common practice when using these algorithms to apply a dimensionality reduction technique as a preprocessing step. However, in addition to the fact that techniques such as Principal Component Analysis (PCA) can only generate linear embeddings, this two-step approach may be sub-optimal as the assumptions underlying the dimensionality reduction and the clustering techniques are independent. A better solution would be an integrated approach generating an embedding specifically optimised for the purpose of clustering.

Such a solution can be found in the field of deep clustering, where recent techniques have been developed to train a deep neural network in order to learn suitable representations for clustering. This is done by incorporating appropriate criteria and assumptions during the training process, e.g. by introducing new regularisation terms. The ultimate goal is to automatically perform feature extraction, dimensionality reduction and clustering with a single and same model rather than doing these tasks independently with separated models. These deep-learning based approaches have also the advantage of being scalable as they have been developed to deal with big data.

As far as we know, none of the deep clustering algorithms available in the literature have been applied yet to the specific problem of trajectory clustering for air flow identification and characterisation. The purpose of this paper is to explore some of the advantages and inconveniences of such novel approaches and illustrate their use for trajectory and flow analysis.

This study is based on a dataset of 19,480 trajectories landing at Zurich airport. We use autoencoding neural networks to generate a low-dimensional latent space so that we can study some of its inherent properties. In particular, we show that although some clusters are naturally formed in the latent space, there are cases where the points are not well separated and overlap. Finally, we apply a deep clustering technique to demonstrate that such techniques can generate a latent space with a better separation of the clusters. The impact of these techniques on the way trajectories are actually clustered into flows will be discussed. We begin with a literature review in Section II, present the dataset and our methodology in Section III, then we provide a comparative analyses of the results in Section IV.

II. LITERATURE REVIEW

A. Trajectory clustering

Clustering is an unsupervised data analysis technique widely used to group similar entities into clusters according to a similarity (or distance) function. Multiple clustering algorithms exist in the literature to cluster point-based data such as k-means [1], BIRCH [2], OPTICS [3], DBSCAN [4] or H-DBSCAN [5]. When clustering is applied to trajectories, it requires a proper distance function to be defined between trajectory pairs, which is challenging because of the functional nature of trajectories. The most common approach is to simply sample the trajectory to obtain a n-dimensional vector of points for the use of point-based clustering algorithms and distances such as the Euclidean one.

Other distances exist to better take into account the geometry of the trajectories and in particular their shape: the best well known shape-based distances are Hausdorff [6] and Fréchet [7]. More recently, a more promising shape-based called Symmetrized Segment-Path Distance (SSPD) distance has been proposed [8], [9] which takes into consideration several trajectory aspects: the total length, the variation and the physical distance between two trajectories.

A significant number of clustering methods exist in the literature for flow identification, where the goal is to determine the set of clusters that best fit the operational air traffic flows within an airspace. These methods associate each cluster to an air traffic flow which is a traffic pattern with both temporal and spatial characteristics. The exact definition of what a flow is ultimately depends on the specific application context. Also, it is worth noticing that a majority of the research have focused primarily on studying the spatial dimension of flows.

Some flow identification methods are specifically applied to terminal area operations, which is also the focus in our paper. For instance, Eckstein [10] combines PCA with k-means to evaluate the performance of individual flights in TMA procedures. Rehm [11] and Enriquez [12] apply hierarchical and spectral clustering techniques to identify air traffic flow patterns from and to an airport. Murça et al. [13], [14] present a framework based on DBSCAN and other machine learning algorithms to identify and characterise air traffic flow patterns in terminal airspaces. Olive et al. [15] propose a specific clustering technique for identifying converging flows in terminal areas, which helps understand how approaches are managed.

B. Deep clustering

The ever increasing amount of generated ADS-B data offers an unprecedented opportunity to the ATM research community for knowledge discovery in trajectory data through the application of unsupervised clustering algorithms.

Unfortunately, conventional clustering algorithms present serious performance issues when applied to high-dimensional large-scale datasets. Prior to the application of the clustering algorithm, the features need to be manually and carefully extracted from data and a dimensionality reduction technique applied. Solutions to all these issues have been at the core of the success in the deep learning field, even though the application of deep neural networks to the specific problem of clustering (deep clustering) is relatively more recent and less well-known.

Deep clustering is however an active area of research with a number of methods already published [16], [17]. Even though it is out of the scope of the paper to provide a detailed review of the field, we will introduce the main principles and provide a focus on the main references on autoencoder-based deep clustering.

Deep clustering relies on a deep neural network to learn a representation or embedding of the input data (e.g. trajectories) into a lower-dimensional latent space by minimising the following loss function:

$$L = L_n + \lambda L_c \tag{1}$$

where L_n is the network loss, L_c is the clustering loss and λ is a hyper-parameter to control the weight of the clustering loss. The network loss ensures that the learned embedding respects the structure of the original data preventing the network from finding trivial solutions or corrupting the latent space. The clustering loss constrains the points in the latent space to become more separated or discriminative.

Autoencoder-based deep clustering is probably the most popular approach and the methods in this category can use either the basic autoencoder or any of its multiple variants (convolutional autoencoder, denoising autoencoder, sparse autoencoder) to define the network architecture. In this approach, the reconstruction error corresponds to the network loss whereas the clustering loss is specific to the method: cluster assignment loss in both the Deep Embedded Clustering (DEC) [18] and Deep Convolutional Embedding Clustering (DCEC) [19] or the k-means loss in the Deep Clustering Network (DCN) [20].

In spite of the previously mentioned advantages of the deep clustering approach, there are also some important challenges to be considered. First, the optimal setting of the hyper-parameters is not obvious, especially in an unsupervised setting where there is no real validation data to be used as guidance. Secondly, the lack of interpretability intrinsic to the use of neural networks is even more of an issue because there is no validation data on which we could build a certain level of trust. Lastly, most of the deep clustering methods lack the theoretical ground needed to estimate the generalisation of their performance.

III. Methodology

A. Reference dataset

The methodology is tested with a dataset including a total of 19,480 trajectories landing at Zurich airport (`LSZH`) between October 1st and November 30rd 2019. We relied on The OpenSky Network [21] database to properly label trajectories landing at `LSZH`.

All trajectories have been requested and preprocessed with the help of the `traffic` Python [22] library which downloads OpenSky data, converts the data to structures wrapping `pandas` data frames and provides a specialised semantics for aircraft trajectories (e.g., intersection, resampling or filtering). In particular, it iterates over trajectories based on contiguous

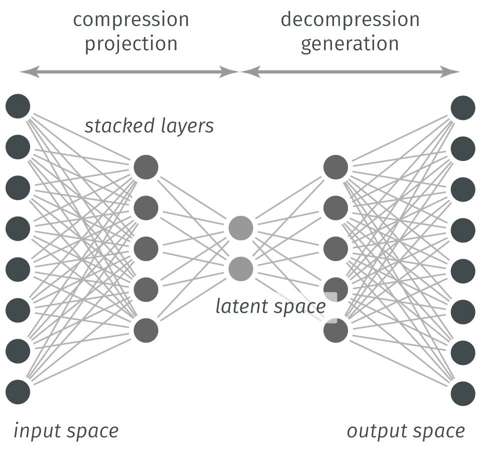

Figure 1. Structure of an autoencoding neural network

timestamps of data reports from a given icao24 identifier: all trajectories are first assigned a unique identifier and resampled to one sample per second before the bearing of the initial point within 40 nautical miles (ASMA area) is computed.

The data presented in this paper is now provided as a generic import in the traffic library, triggering a download from a corresponding figshare repository [23] if the data is not in the cache directory of the user.

B. Autoencoding neural networks for anomaly detection

Autoencoders are artificial neural networks consisting of two stages: encoding and decoding. Autoencoders aim at finding a common feature basis from the input data. They reduce dimensionality by setting the number of extracted features to be less than the number of inputs. Autoencoder models are usually trained by backpropagation in an unsupervised manner.

The encoding function of an autoencoder maps the input data $s \in \mathbb{R}^d$ to a hidden representation $y \in \mathbb{R}^h = e(s)$. The decoding function maps the hidden representation back to the original input space according to $\hat{s} = d(y)$.The objective of the autoencoder model is to minimise the error of the reconstructed result:

$$(w,b,w',b') = \operatorname{argmin} \ell(s, d(e(s))) \quad (2)$$

where $\ell(u,v)$ is a loss function determined according to the input range, typically the mean squared error (MSE) loss:

$$\ell(u,v) = \frac{1}{n} \sum ||u_i - v_i||^2 \quad (3)$$

Figure 1 shows an example of autoencoding neural network structure where layers are stacked so as to better handle volume and complexity in data sets we analyse. Autoencoders (also referred to in a wider sense as *reconstruction methods*) have proven useful at detecting anomalies in large data sets, and in aircraft trajectories [24], [25], [26].

C. Information extraction on the latent space

Figure 1 shows how the whole dataset projects onto a two-dimensional latent space through the first part of the autoencoding neural network (a projection operator). Samples are projected onto a smaller dimension space before a generation operator attempts at reconstructing the original samples.

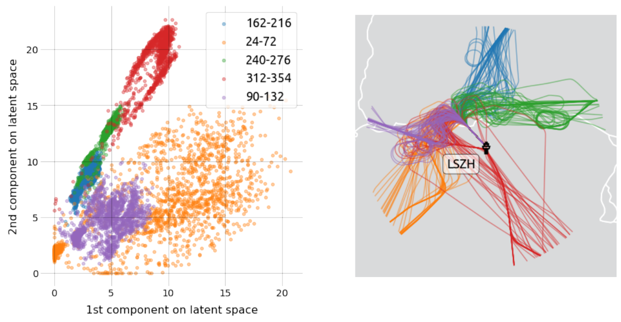

Figure 2. Trajectories landing at LSZH airport, clustered using the bearing angle when entering the ASMA area on runway 14, within 40 nautical miles from the airport.

In spite of a poorer reconstructive power, we projected our samples down to a two-dimensional space in order to plot the distribution of projected samples on Figure 2. Here we assigned colours to each trajectory based on the bearing with respect to the airport at the moment the aircraft enter a 40 nautical miles radius: each colour is associated to an incoming flow into the TMA.

Figure 3 restricts the dataset to trajectories which are self-intersecting (holding patterns) and assigns a colour to trajectories landing from a bearing angle between 90 and 132 degrees. Three pairs of trajectories which are close to each other in their two-dimensional latent representation are plotted: the top right map display two trajectories stacking two holding patterns before getting the clearance to land on runway 14. The two other pairs also display similar features when plotted on a map.

Looking a the latent space of autoencoders trained to reconstruct trajectories is particularly enlightening with respect to their ability to extract information from large amounts of data which comes as a side of effect of their first intended use.

D. Enforcing a clustered structure in the latent space

As autoencoders seem to naturally separate similar data in the latent space, we present in this section two methods to further enforce a clustering of high-dimensional data based on their representation on a lower dimensional space: artefact (AutonecodeR-based t-SNE for AirCraft Trajectories) and DCEC (Deep Convolutional Embedded Clustering) [19].

First, we developed the artefact method which is based on t-SNE (t-distributed Stochastic Neighbour Embedding) [27] as a way to force a more fitted representation in the latent space for the traditional clustering methods. Second, we adapted an existing deep clustering method named DCEC, which was originally designed for images, so it can be applied to the clustering of trajectories.

The goal is to compare these two methods in terms of clustering capabilities. The advantage of DCEC over artefact is that it can identify the clusters directly without the need of using a classical clustering method. However, DCEC needs

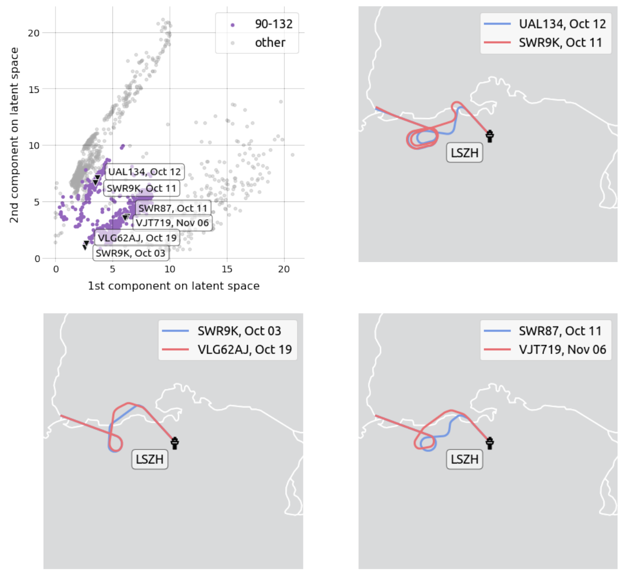

Figure 3. The scatter plot displays the latent space on trajectories containing one or several loops. Sample that are close to each other reveal similar patterns on the map. In particular, trajectories stacking holding patterns are plotted on the top right in purple and the subset of trajectories corresponding to strictly more than one holding pattern are located in the top left purple cluster.

the number of clusters to be set in advance, which is not necessarily the case with artefact if used in combination with algorithms like DBSCAN.

1) artefact: This method is based on an autoencoder and a loss function combining the reconstruction error and a regularisation term based on t-SNE.

t-SNE [27] is a dimensionality reduction method and also a powerful 2D or 3D visualisation technique for high-dimensional data. The projection operator is not applicable to new samples of data; yet this technique is commonly used as a hint to determine whether samples would be separable for classification with a (deep) artificial neural network.

Given a set of n-dimensional samples $\{x_i \in X\}$, t-SNE defines a conditional probability $p_{j|i}$ between two samples x_i and x_j (proportional to the similarity between x_i and x_j) as the probability for x_j to be a neighbour of x_i given a Gaussian distribution of x_j samples around x_i:

$$p_{j|i} = \frac{e^{-d(x_i,x_j)^2/2\sigma_i^2}}{\sum_{k\neq i} e^{-d(x_k,x_i)^2/2\sigma_i^2}} \tag{4}$$

and a distribution of similarity between x_i and x_j as:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \tag{5}$$

t-SNE also defines a similarity distribution on the latent space based on a heavy-tailed Student t-distribution (one-degree of freedom) defined as:

$$q_{ij} = \frac{\left(1+d(z_i,z_j)^2\right)^{-1}}{\sum_{k\neq l}\left(1+d(z_k,z_l)^2\right)^{-1}} \tag{6}$$

t-SNE tends to minimise similarities (defined in terms of Kullback-Leibler distances) among distances in the large dimension space X and in the latent space Z:

$$d_{KL}(P||Q) = \sum_{i,j} p_{ij} \log\left(\frac{q_{ij}}{p_{ij}}\right) \tag{7}$$

2) DCEC: This method is an evolution of DEC [18] and uses a convolutional autoencoder instead of a feed-forward autoencoder. Unlike in artefact, the goal of DEC/DCEC is not the minimisation of the KL divergence to generate a latent structure representative of the distances between data points in the original data space. Instead, DEC/DCEC defines a centroid-based probability distribution and minimises its KL divergence to an auxiliary target distribution to simultaneously improve clustering assignment and feature representation. This centroid-based method reduces complexity from $O(n^2)$ in artefact to $O(nk)$, where k is the number of centroids. However, this requires the number of clusters to be fixed in advance.

More precisely, both DEC/DCEC use the cluster assignment hardening loss [17], [18]. First, the soft assignment of points to clusters is done via the Student's t-distribution with one-degree of freedom. which is a kernel measuring the similarity between the points in the latent space and the cluster centroids:

$$q_{ij} = \frac{\left(1+d(z_i,u_j)^2\right)^{-1}}{\sum_{j'}\left(1+d(z_i,\mu_{j'})^2\right)^{-1}} \tag{8}$$

where z_i is a point in the latent space and μ_j is the j-th cluster centroid and q_{ij} represents the probability of point z_i to be assigned to cluster j. Note that in t-SNE this distribution is calculated from the pairwise distance between points in the latent space.

The auxiliary target distribution in DCEC is the one proposed in DEC and is defined as:

$$p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}} \tag{9}$$

where $f_j = \sum_i q_{ij}$ are the soft cluster frequencies. By squaring the the original Q distribution and then normalising it, P forces the soft assignments to become closer to 0 or 1. The divergence between these two distributions is computed via the KL divergence as in Equation (7).

E. Clustering in the latent space

The training process of both artefact and DCEC consists in the minimisation of the following loss function which is based on both the network loss (reconstruction error) and the clustering loss (KL divergence):

$$(w,b,w',b') = \operatorname{argmin} \ell(s,d(e(s))) + \lambda d_{KL}(P||Q) \tag{10}$$

However, the two clustering losses are in fact of a very different nature, each one being an example of the two categories identified by Min et al. [16]: auxiliary clustering loss and principal clustering loss.

An *auxiliary clustering loss* only guides the network to obtain a more clustering-friendly embedding from which the clusters have to be identified by running a clustering algorithm after the network training process. The t-SNE regularisation term in artefact plays exactly this role, with the actual clustering being performed in the fitted latent space by a separated algorithm such as DBSCAN or Gaussian mixture.

On the other hand, a *principal clustering loss* allows to obtain directly the cluster assignments after the training process. This is the case of the cluster assignment hardening loss used by DEC/DCEC where the cluster assignment j of each sample i can be obtained directly from distribution Q from $\mathrm{argmax_j}(q_{ij})$.

F. Application and implementation

Trajectories are downsampled with a rate of 64 points per trajectory and four features per trajectory point have been selected for clustering: true track angle (unwrapped), longitude, latitude and altitude. We focused on two specific subsets of trajectories: all trajectories landing on runway 14 (14,441 trajectories) and a focus on the subset of trajectories coming from a northbound flow (initial bearing between 162 and 216°, 4437 trajectories), specifically addressed in Section IV-C. For the sake of reproducibility, we provide below a declarative description of our preprocessing using Python `traffic` [22] library (version 2.3):

```python
landing_zurich_2019
.resample(64)  # unwrap angles and reduce to 64 samples
.query("runway == '14' and initial_flow != 'N/A'")
.query("initial_flow == '162-216'")  # 2nd dataset only
.eval()  # triggers iteration, evaluation and reduce
```

As determining hyper-parameters by cross-validation is not possible in unsupervised learning, we set up them only guided by the visualisation of results.

artefact uses a feed-forward autoencoder with network dimensions set to d-32-8-h for the encoder (decoder is a mirror of the encoder), where d is the data-space dimension (64 samples/trajectory $\times 4$ features) and $h = 2$. Batch size is equal to the dataset length when $\lambda = 0$ and 1000 otherwise. The number of iterations per batch is set to 800 if $\lambda > 0$ and 5000 otherwise. The auxiliary clustering algorithm is a Gaussian mixture with the number of components set to 5 for the whole traffic dataset and to 4 for the other two smaller datasets.

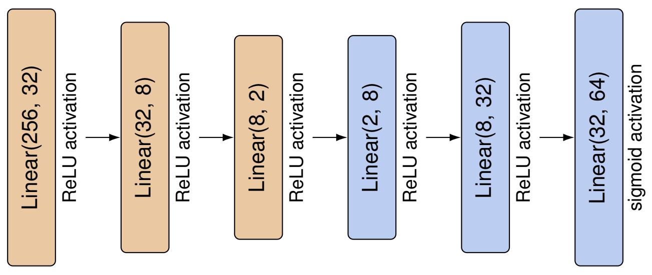

Figure 4. Architecture implemented for the artefact autoencoder

The convolutional autoencoder architecture from DCEC has been adapted to take into account the smaller dimensionality

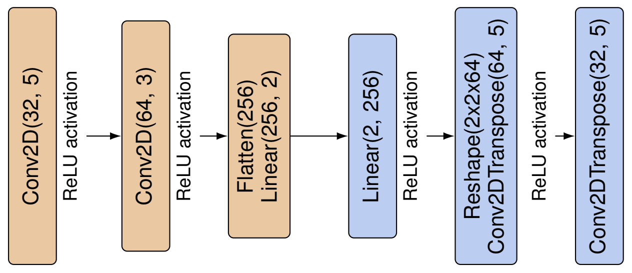

Figure 5. Architecture implemented for the DCEC convolutional autoencoder

of trajectories when compared to the one of the images used in DCEC. Thus, we have removed one of three original convolutional layers in the encoder (as well as the corresponding deconvolutional layer in the decoder) and the embedding dimension h is set to 2 like in artefact. The resulting architecture is shown in Figure 5, where `Conv2d(n, k)` (and its transpose `Conv2DTranspose(n, k)`) denotes a convolutional (deconvolutional) layer with n filters and kernel size $k \times k$. The stride length is set to 2 in all convolutional layers. The input trajectories are reshaped from 64×4 to $8 \times 8 \times 4$ to work with the 2D convolutional layers. The number of clusters is set to 5 for the dataset with the full traffic and 4 otherwise. The convolutional autoencoder is pre-trained (without the clustering layer) for 1000 epochs and then fine-tuned with the clustering layer enabled for 1000 iterations per batch. Batch size is set to 1000 samples.

IV. Results and Analysis

We trained several clustering models using both artefact and DCEC with several λ values in order to assess the effect of the clustering loss parameters introduced in III-E. The $\lambda = 0$ executions serve as baseline for vanilla (convolutional) autoencoders. Then, we trained models on a subset of trajectories which is already commonly identified as a flow from an operational perspective using inbound flow and runway.

A. Baseline for comparison: $\lambda = 0$

In order to properly assess the benefits of the λ term (i.e. of the impact of regularisation terms), we introduce in this section baseline results with $\lambda = 0$ on both artefact (i.e. simple autoencoder) and DCEC (i.e. convolutional autoencoder).

Figure 6 shows the latent space structure as generated by artefact (simple autoencoder). Inbound flows are properly separated, and even further clustered. Figure 7 confirms the idea that autoencoders display an intrinsic separating ability on their latent space: we selected two inbound flows and coloured trajectories on the map with respect to clusters emerging on their latent space. We kept original colours for the most direct trajectories and complementary colours (i.e. orange vs. blue; red vs. green) for the secondary emerged clusters, roughly corresponding to stacked holding patterns.

Figure 8 shows the latent space structure as generated by DCEC (convolutional autoencoder) on the same subset of trajectories. Samples seem to self organise differently: while clusters organisation seems radial on Figure 6 (or even

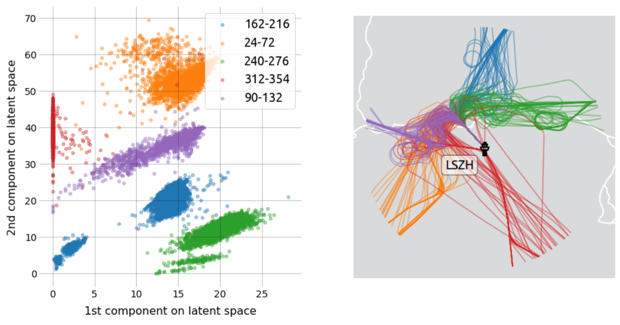

Figure 6. artefact with $\lambda = 0$: this is equivalent to a simple autoencoder.

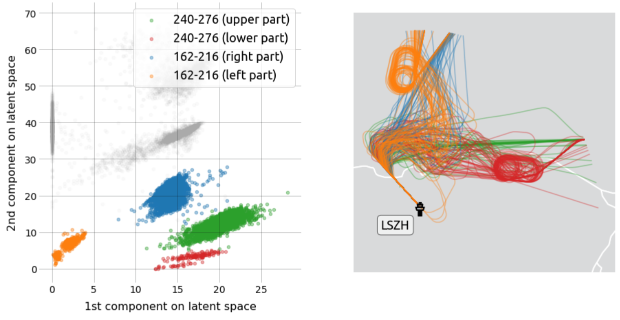

Figure 7. artefact with $\lambda = 0$: focus on how inbound flows are separated.

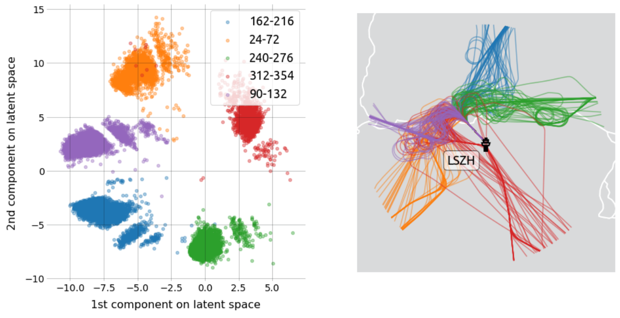

Figure 8. DCEC with $\lambda = 0$: this is equivalent to a convolutional autoencoder.

Figure 2 which was trained only on the track angle feature), whereas samples seem to distribute in a more scattered pattern on Figure 8. Although, to our knowledge, no scientific proof supports this description, we found these distribution patterns on 2D latent spaces consistent enough across executions and data sets to be mentioned in these lines.

B. Structure of the latent space with clustering loss

In this section we look into the effect of the clustering loss on the structuring of the latent space. Colours are still to be related with the naive flow identification (incoming flow – runway) in order to help describing how the latent space structures with $\lambda \neq 0$.

Figure 9 compares the structure of the latent space constructed with artefact $\lambda = 0$ (equivalent to Figure 6) and $\lambda = 0.5$, then plots the corresponding trajectories on a map. For the sake of readability, we never plot more than 50 trajectories per cluster. The first line of the grid plots the latent

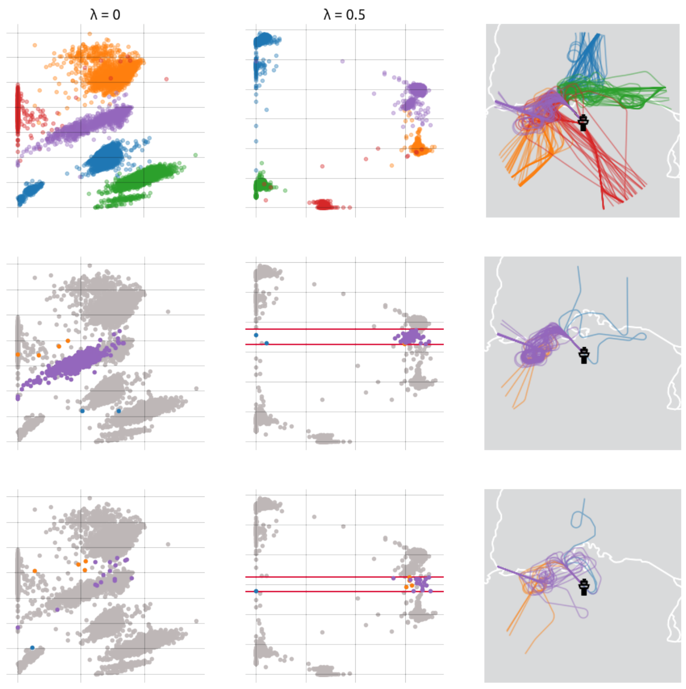

Figure 9. Comparison of latent spaces for artefact with $\lambda = 0$, $\lambda = 0.5$, and corresponding samples on the right. Colours are consistent with initial flows in the beginning of the paper.

space for the whole dataset: we can observe that samples have been further pushed away from each other in a manner that is consistent with the incoming flows.

In the second and third lines of the grid, we focus on specific samples selected in the latent space of the $\lambda = 0.5$ training, between the two red lines. Selected samples are then highlighted in the latent space generated with $\lambda = 0$ (the left column of the grid).

The second line selects a subset of purple trajectories (flow from the North-West) which are mostly stacking holding patterns in a predetermined position. Few orange samples seem to overlap as they also stack holding patterns in the same location. On the left side of the latent space, the selection caught two outliers from the blue set of trajectories (flow from the North) which are obviously non standard.

The third line shifts the selection interval below the one depicted on the second line and selects more orange trajectories stacking holding patterns; also, few purple trajectories are selected as they cross the final approach before aligning to runway 14 on a left-hand turn. On the left side of the latent space, another outlier from the Northern flow has been selected, which lands in Zurich airport after three attempts.

Figure 10 implements a similar comparison with the latent space generated with DCEC. Similarly, clusters are further pushed away from each other with $\lambda = 0.5$ and some trajectories from different initial flows become closer to each other: we can observe the emergence of three clusters grouping trajectories from different initial flows.

The second line of the grid displays the cluster resulting

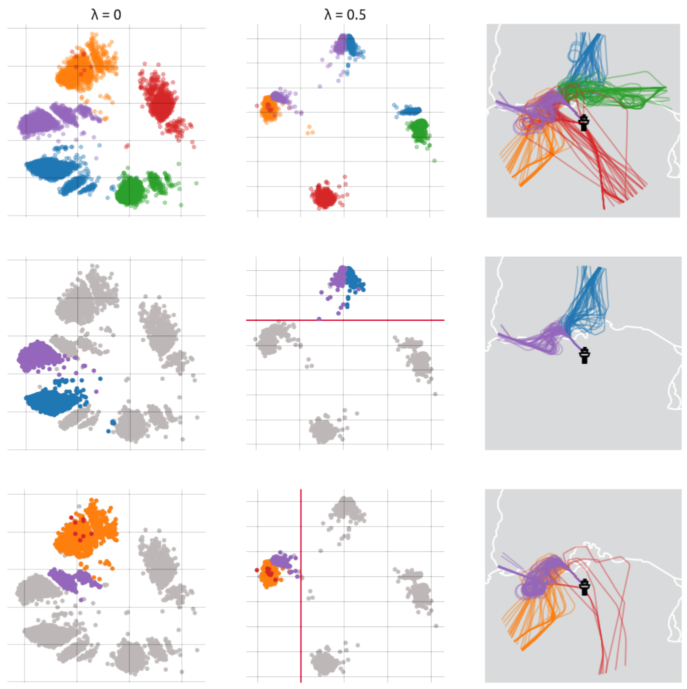

Figure 10. Comparison of latent spaces for DCEC with $\lambda = 0$, $\lambda = 0.5$, and corresponding samples on the right. Colours are consistent with initial flows in the beginning of the paper.

from the merging of two incoming flows with no holding pattern. Interestingly, it seems that the neural network captured the symmetry in the two flows converging to runway 14. The third line of the grid grouped purple and orange trajectories stacking holding patterns. With $\lambda = 0$, DCEC grouped all trajectories coming from the North-West in a single cluster (arguably separated by a thin empty space); with $\lambda = 0.5$, direct and tromboning trajectories are more clearly separated from holding trajectories.

C. Hierarchical clustering

In this section, we use artefact and DCEC in order to perform a clustering on the latent space that is trained. DCEC comes with its own clustering method; we perform a separate Gaussian Mixture on the latent space constructed by artefact. After incoming flows are separated by a first clustering, we recursively apply a new clustering on each identified flow.

We implemented artefact and DCEC on the Northern flow (between 162 and 216°) with a hyperparameter of 4 clusters to identify. Figure 11 plots the four constructed latent spaces with artefact and DCEC, for $\lambda = 0$ (baseline) and $\lambda = 0.5$.

Figure 12 display the identified clusters with artefact. Results are similar for the green (holding patterns and tromboning) and orange clusters. However, we observe a transfer of trajectories from the red cluster to the blue cluster: with $\lambda = 0.5$, the two branches to the North of the ASMA area are separated: the Eastern branch is grouped in the red cluster, while the Western branch is further separated between (almost) direct flights to the final approach fix (orange cluster) and trajectories with little tromboning (blue cluster).

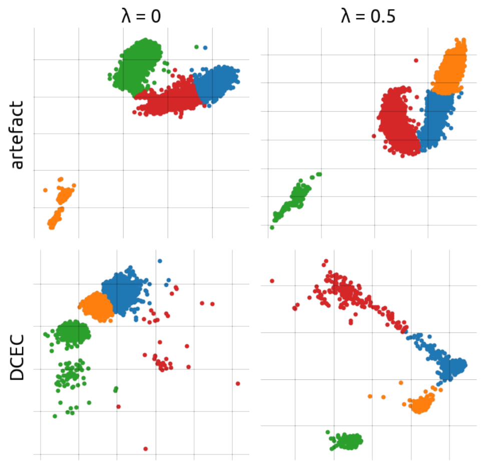

Figure 11. Latent spaces for flow 162-216 generated by artefact and DCEC with $\lambda = 0$ and $\lambda = 0.07$

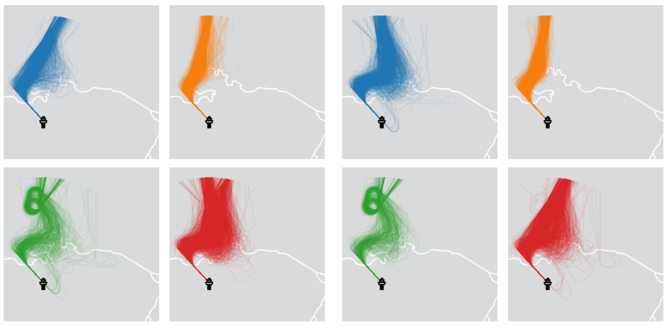

Figure 12. Clusters identified by artefact for one flow with $\lambda = 0$ (left) and $\lambda = 0.5$ (right); to be related with the first line of the grid in Figure 11

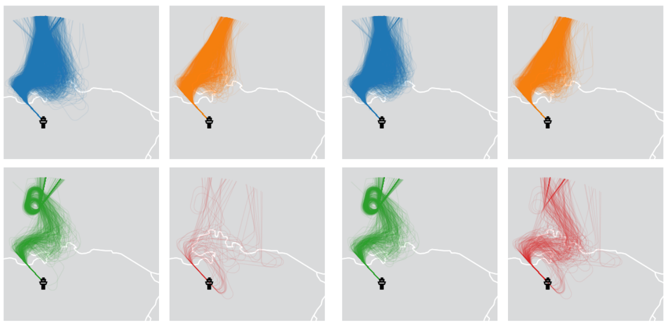

Figure 13. Clusters identified by DCEC for one flow with $\lambda = 0$ (left) and $\lambda = 0.5$ (right); to be related with the second line of the grid in Figure 11

Figure 13 display the identified clusters with DCEC. Identified clusters are different with this implementation. The most notable impact of the clustering loss is the transfer of tromboning trajectories, mostly from the blue cluster to the red cluster which was almost empty with $\lambda = 0$.

Although artefact and DCEC result in slightly different clusters, both provide a precious insight about the organisation of traffic under different complexity conditions. Both methods separate trajectories stacking holding patterns, and manage to isolate different patterns, including direct or trombones.

V. Conclusions

Traditional clustering techniques based on similarity or distance functions defined in a high-dimensional data space are ineffective and do not scale well with large datasets. In this paper, we have studied the application of clustering techniques developed in the field of deep learning to the problem of air traffic flow identification. These techniques provide automatic feature extraction and dimensionality reduction by generating a low-dimensional latent space which is more fitted to the clustering task. This is done by integrating into the loss function a regularisation term called clustering loss.

We presented and compared two autoencoder-based trajectory clustering methods: artefact and DCEC. We applied both methods to identify the arrival flows in the Zurich terminal area and shown how they can provide an embedding of the trajectories which is more discriminative for clustering. Although both methods are able to identify the arrival flows, they present some functional and performance differences due to the nature of their clustering losses. While artefact needs to be combined with a clustering algorithm such as a GMM or DBSCAN to identify the clusters, DCEC can do so directly. However, DCEC always needs the number of clusters to be fixed a priori, which is not the case for artefact if used with DBSCAN for instance.

One of the main issues with deep clustering methods is the significant number of hyper-parameters that needs to be fitted. In our study, we mainly focused on the impact of the λ parameter on the results. However, we neither explored the influence nor optimised the values of other parameters. In particular, for the sake of simplicity and illustration purposes, we have systematically fixed the dimension of the latent space to 2, even though a higher dimension could have been more appropriate. More generally, the difficulty stems from the lack of validation data which hampers the evaluation process of unsupervised methods.

Both clustering methods presented in this paper can also be used for trajectory anomaly detection. An anomaly score for each trajectory can be computed from either the reconstruction error or the probability of cluster membership when artefact is combined with a model like a Gaussian mixture or directly from DCEC Q distribution. However, the definition of an appropriate threshold is required to label each trajectory as anomalous or not. Future works will assess the impact of clustering losses on the performance of reconstruction-based anomaly detection methods.

References

[1] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[2] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in *ACM Sigmod Record*, vol. 25, no. 2, 1996.

[3] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: ordering points to identify the clustering structure," in *ACM Sigmod record*, vol. 28, no. 2. ACM, 1999, pp. 49–60.

[4] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*, vol. 96, no. 34, 1996, pp. 226–231.

[5] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2013, pp. 160–172.

[6] F. Hausdorff, *Grundzuge der Mengenlehre*. American Mathematical Society, 1978, vol. 61.

[7] M. Fréchet, "Sur quelques points du calcul fonctionnel," *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, vol. 22, no. 1.

[8] P. Besse, B. Guillouet, J.-M. Loubes, and R. François, "Review and perspective for distance based trajectory clustering," *arXiv preprint arXiv:1508.04904*, 2015.

[9] B. Guillouet, "Apprentissage statistique: application au trafic routier à partir de données structurées et aux données massives," Ph.D. dissertation, Université Toulouse III – Paul Sabatier, 2016.

[10] A. Eckstein, "Automated flight track taxonomy for measuring benefits from performance based navigation," in *2009 Integrated Communications, Navigation and Surveillance Conference*. IEEE, 2009, pp. 1–12.

[11] F. Rehm, "Clustering of flight tracks," in *AIAA Infotech@ Aerospace 2010*, 2010, p. 3412.

[12] M. Enriquez, "Identifying temporally persistent flows in the terminal airspace via spectral clustering," in *Tenth USA/Europe Air Traffic Management Research and Development Seminar (ATM 2013)*, 2013.

[13] M. C. R. Murça, R. DeLaura, R. Hansman, R. Jordan, T. Reynolds, and H. Balakrishnan, "Trajectory clustering and classification for characterization of air traffic flows," *AIAA Aviation*, 2016.

[14] M. C. R. Murça, R. J. Hansman, L. Li, and P. Ren, "Flight trajectory data analytics for characterization of air traffic flows: A comparative analysis of terminal area operations between New York, Hong Kong and Sao Paulo," *Transportation Research Part C: Emerging Technologies*, vol. 97, pp. 324–347, 2018.

[15] "Trajectory clustering of air traffic flows around airports," vol. 84.

[16] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A survey of clustering with deep learning: From the perspective of network architecture," *IEEE Access*, vol. 6, pp. 39 501–39 514, 2018.

[17] E. Aljalbout, V. Golkov, Y. Siddiqui, and D. Cremers, "Clustering with deep learning: taxonomy and new methods (2018)," *arXiv preprint arXiv:1801.07648*, 1801.

[18] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*, 2016, pp. 478–487.

[19] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep clustering with convolutional autoencoders," in *International Conference on Neural Information Processing*. Springer, 2017, pp. 373–382.

[20] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017, pp. 3861–3870.

[21] M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, and M. Wilhelm, "Bringing up OpenSky: A large-scale ADS-B sensor network for research," in *Proceedings of the 13th international symposium on Information processing in sensor networks*, 2014, pp. 83–94.

[22] X. Olive, "traffic, a toolbox for processing and analysing air traffic data," *Journal of Open Source Software*, vol. 4, no. 39, p. 1518, Jul. 2019. [Online]. Available: http://joss.theoj.org/papers/10.21105/joss.01518

[23] X. Olive and L. Basora, "Reference data sets for detection and identification of significant events in historical aircraft trajectory data." Dec 2019. [Online]. Available: https://doi.org/10.6084/m9.figshare.11406735.v1

[24] X. Olive, J. Grignard, T. Dubot, and J. Saint-Lot, "Detecting Controllers' Actions in Past Mode S Data by Autoencoder-Based Anomaly Detection," in *Proceedings of the 8th SESAR Innovation Days*, 2018.

[25] X. Olive and L. Basora, "Identifying Anomalies in past en-route Trajectories with Clustering and Anomaly Detection Methods," in *Proceedings of the 13th USA/Europe Air Traffic Management Research and Development Seminar*, 2019.

[26] L. Basora, X. Olive, and T. Dubot, "Recent advances in anomaly detection methods applied to aviation," *Aerospace*, vol. 6, no. 11.

[27] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.