Expert Systems With Applications 198 (2022) 116733

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

# Deep flight track clustering based on spatial–temporal distance and denoising auto-encoding

Guoqian Liu [a,b], Yuqi Fan [a,b], Jianjun Zhang [a], Pengfei Wen [a], Zengwei Lyu [a], Xiaohui Yuan [c,*]

[a] Hefei University of Technology, Hefei, 230009, China
[b] Key Laboratory of Knowledge Engineering with Big Data (Hefei University of Technology), Ministry of Education, Hefei, 230009, China
[c] University of North Texas, Denton, TX, 76203, USA

ARTICLE INFO

*Keywords:*
Clustering
Similarity
Denoising auto-encoding
Spatial
Temporal

ABSTRACT

The rapid development of the aviation industry imposes an urgent need for airspace traffic management. Meaningful clustering of flight tracks is of paramount importance for efficient operation and management of increasingly complex aerial space and traffic. Two key components exist in track clustering: similarity metric and clustering method. Most of the existing studies on track similarity metrics only consider the spatial coordinates of the track points without taking into consideration of the rich information of the track data, such as flight heading and flight speed, on the measurement of track similarity. In addition, temporal properties and the derived features of the flight tracks shall be utilized to reveal the underlying patterns and overcome distortions from noise. In this paper, we propose a track similarity based on the spatial–temporal characteristics of flight tracks and a Deep Temporal Clustering method using a denoising autoencoder. Our proposed method employs the Deep Temporal Denoising Auto-encoding network to extract the latent representations of the track sequences. By extending the idea of $k$-means clustering, Deep Temporal Clustering groups the flight tracks with a Time Clustering Layer. Experiments are conducted using Automatic Dependent Surveillance-Broadcast track data. In comparison with classical and state-of-the-art methods, among all cases, our Deep Temporal Clustering method achieved a much-improved performance of more than 57.3%. When we introduce noise to the track records and increase its magnitude, the performance of our method degrades but the trend slows down as the noise magnitude increases. The change is less than 7% and, in some cases, is close to zero, which demonstrates the robustness of our method to noise.

## 1. Introduction

The aviation industry has been steadily advancing with many airlines and flights among cities worldwide, which imposes an urgent need for airspace traffic management. With the increasing air traffic, the flight trajectory data increases exponentially. The study of flight track clustering is the premise and foundation of air traffic control, which identifies the traffic patterns from flight track data and finds clusters according to operational flows within an airspace (Oliver, Basora, Viry, & Alligier, 2020). For sector planning, clustering is performed on the historical tracks, which enables us to obtain sector boundaries. In addition, fight clusters allow us to generate the average flight track, which enables us to gain improvements in flight operations. In the terminal area, the cluster analysis of flight trajectories helps decision-making in airspace scheduling and management. Meaningful clustering of flight tracks is of paramount importance for efficient operation and management of increasingly complex aerial space and traffic.

Track clustering can be achieved via analyzing track points, grouping sub-tracks, and evaluating the entire tracks. However, two key components exist in these approaches: (1) a metric for measuring the distance or similarity between flight tracks or points; and (2) an algorithm to divide the tracks into groups. For computing the similarity between track data, most existing methods rely on Euclidean distance, Manhattan distance, Cosine Similarity, Dynamic Time Warping (DTW) distance, Longest Common Sub-Sequence (LCSS) distance, etc. (Dahlbom & Niklasson, 2007; Lee, Han, & Li, 2008; Piciarelli, Foresti, & Snidaro, 2005). With regard to clustering strategies, methods such as $k$-means (Lloyd, 1982), Affinity Propagation (AP) (Liu, Liu, Jin, Li, & Zheng, 2020), density (Pan, He, Wang, Xiong, & Peng, 2016) have been employed. Note that flight track data consist of both spatial and temporal information. Each track point includes information such as the position of the flight and the points in a track form an ordered

* Corresponding author.
*E-mail addresses:* guoqian_liu@mail.hfut.edu.cn (G. Liu), yuqi.fan@hfut.edu.cn (Y. Fan), jianjun@hfut.edu.cn (J. Zhang), 2017111025@mail.hfut.edu.cn (P. Wen), lzw@hfut.edu.cn (Z. Lyu), xiaohui.yuan@unt.edu (X. Yuan).

https://doi.org/10.1016/j.eswa.2022.116733
Received 21 June 2021; Received in revised form 21 February 2022; Accepted 22 February 2022
Available online 5 March 2022

temporal sequence. The temporally and spatially uneven data points introduce an interesting but challenging clustering problem. Most of the existing studies on track similarity metrics only consider the spatial coordinates of the track points without taking into consideration of the rich information of the track data, such as flight heading and flight speed, on the measurement of track similarity. In addition, temporal properties and the derived features of the flight tracks shall be utilized to reveal the underlying patterns and overcome distortions from noise.

In this paper, we propose a method for clustering flight tracks based on the space–time relationship of the trajectory data. The main contributions of this paper are as follows:

- We introduce a novel metric of track similarity based on the spatial–temporal characteristics of flight tracks. The similarity metric integrates the point-to-point distance, point-to-track distance, and ultimately track-to-track distance for characterizing the resemblance of the flight tracks.
- We propose a deep trajectory clustering model, Deep Temporal Clustering (DTC), that leverages a denoising autoencoder to extract the latent representation of the track sequences. By extending the idea of k-means clustering, the DTC model clusters the flight tracks using a Time Clustering Layer (TCL). The proposed network fuses the outputs of the autoencoder and the TCL for improved track clustering.

The rest of the paper is organized as follows. Section 2 reviews the related work with respect to similarity metrics and the existing methods for track clustering. Section 3 starts with a formal definition of the problem followed by a detailed description of our proposed method. Section 4 presents our experimental results and a comparison study with the state-of-the-art methods. Section 5 concludes this paper with a summary.

## 2. Related work

### 2.1. Track similarity measurement

Clustering of flight trajectories is based on a track similarity metric, which is challenging for clustering the flight trajectories with time-series track point sequences. The approaches to quantitatively representing similarity are divided into two categories according to whether using the distance to represent the similarity of two objects.

When the distance is used to measure the similarity between two objects, the smaller distance means the two objects are more similar. Current methods of calculating the distance between trajectories include Euclidean distance, Manhattan distance (Niedermeier & Sanders, 1996), Hanming distance, Chebyshev distance, Mahalanobis distance, etc. For example, Euclidean distance is a popular approach that simply samples the trajectory to obtain a multi-dimensional feature vector of track points for further clustering algorithms.

Some approaches represent similarity between two object via kinds of non-distance functions. Popular such methods include Cosine Similarity (COS) (Zobel & Moffat, 1998), Pearson Correlation-based Similarity (COR) (Golay et al., 1998), Dynamic Time Warping (DTW) (Li, 2021), Longest Common Sub-Sequence (LCSS) (Gariel, Srivastava, & Feron, 2011), divergence, trajectory point or shape similarity, track pattern similarity, multiple feature similarity, etc. (Dahlbom & Niklasson, 2007; Lee et al., 2008; Piciarelli et al., 2005).

For trajectory point or shape similarity, Rehm (2010) proposed a track similarity metric based on track point comparison. Lv, Kang, Lu, and Xu (2021) used pairwise similarity and self-expression layer to weigh the reconstruction loss to capture local structure information for deep clustering. Yang, Tang, Chen, and Wang (2021) defined a similarity metric between two trajectories by considering the starting time and the Euclidean distance between the closest track points of each period of trajectories. Besse et al. proposed a shape-based distance, the Symmetrized Segment-Path Distance (SSPD), which compares trajectories as a whole to avoid incidental variation between trajectories (Besse, Guillouet, Loubes, & Royer, 2016). Seal, Karlekar, Krejcar, and Gonzalo-Martin (2020) proposed a divergence-based similarity measure by using the idea of Jeffrey's divergence. Zhang and Shi (2021) proposed an improved trajectory similarity measure based on SSPD.

For track pattern similarity, Cai, Lee, and Lee (2018) used the improved Ordering Points To Identify the Clustering Structure (OPTICS) algorithm to extract the common behavior pattern of trajectories with additional contextual semantic annotations according to different application scenarios. Liu and Guo (2020) proposed a semantic similarity of trajectories to capture global relationships among trajectories based on community detection from the perspective of the network.

For multiple feature similarity, Yu, Luo, Chen, and Chen (2019) designed a new multi-feature trajectory similarity measure that uses the characteristics of orientation, speed, shape, location, and continuity of each trajectory for clustering. Varlamis et al. (2021) made use of a similarity measure TraFos to compare multiple aspect trajectories (MATs) across each aspect and then combined similarities into a single measure.

Note that flight trajectory data have both spatial and temporal characteristics since each track point indicates the position of the flight and the points in a track are sequential in time. Most of the existing research on track similarity metrics only considers the positions of track points, while failing to accumulate the impacts of the positions and time of tracks, flight heading, and flight speed, etc. on the measurement of trajectory similarity.

### 2.2. Track clustering

There has been extensive research on clustering algorithms. Depending on the difference of clustering implementations, clustering algorithms can be broadly divided into partition-based methods, density-based methods, etc. Partition-based methods include $k$-means (Lloyd, 1982) and its variations, such as $k$-means++, bi-$k$means, $k$-medoids (Krishnapuram, Joshi, & Yi, 1999), etc. Density-based methods (Campello, Moulavi, & Sander, 2013; Ester, Kriegel, Sander, Xu, et al., 1996; Pan et al., 2016) include Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) (Zhang, Ramakrishnan, & Livny, 1996), OPTICS (Ankerst, Breunig, Kriegel, & Sander, 1999), Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al., 1996), Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) (Campello et al., 2013), etc. Other clustering algorithms include Affinity Propagation (AP) (Liu et al., 2020), Self Organizing Maps (SOM) (Qu et al., 2021), spectral clustering, graph clustering (Kang, Lu, Liang, Bai, & Xu, 2020; Lin, Kang, Zhang, & Tian, 2021), etc.

Some methods of track clustering are dedicated to flight track clustering from the perspective of specific track characteristics. Olive et al. proposed a specific clustering method to identify converging flows in terminal areas (Olive & Morio, 2019). Oxenham et al. proposed a skywave over-the-horizon radar track clustering method based on hypothesis test (Oxenham, 2000). Zhong et al. proposed a clustering algorithm on trajectory characteristic points to better analyze the TMA system (Zhong, Liu, & Qi, 2021). Sun et al. proposed a clustering method based on Minimum Bounding Rectangle and buffer similarity to improve trajectories clustering and reduce computation time (Sun & Wang, 2021).

Some research on track clustering improves classical clustering methods to solve flight track clustering problems. Eckstein combined Proper Orthogonal Decomposition (POD) with $k$-means to evaluate the performance of individual flights in Terminal Control Area (TMA) procedures (Eckstein, 2009). Rehm and Enriquez et al. conducted clustering on the arrival procedures of different runways in the terminal area of an airport based on spectral clustering (Enriquez, 2013; Rehm,

2010). Murca et al. presented a framework based on DBSCAN and other machine learning algorithms to identify and characterize air traffic flow patterns in terminal area (Conde Rocha Murca et al., 2016; Murça, Hansman, Li, & Ren, 2018). Yan et al. proposed a clustering algorithm based on fitness proportionate sharing to map the problem into a multimodal optimization problem (Yan et al., 2019). Gui et al. presented a semantic-based trajectory clustering method for arrival aircraft, which is based on $k$-means and DBSCAN (Gui, Zhang, & Peng, 2021). Devarakonda et al. introduced a clustering approach based on hybrid optimization technique and $k$-means algorithm (Devarakonda, Saidala, & Kamarajugadda, 2021). Wang et al. proposed a trajectory clustering method based on HDBSCAN, which adaptively clusters trajectories with their shape characteristics (Wang, Chen, Chen, & Mou, 2021).

## 3. Deep temporal clustering based on denoising autoencoder

Let $T$ denote a set of flight tracks:

$$T = \{T_1, T_2, \dots, T_N\} \tag{1}$$

where $N$ is the total number of tracks. A track $T_i$ consists of a number of track points:

$$T_i = \{P_{i1}, P_{i2}, \dots, P_{iL}\} \tag{2}$$

where $L$ is the number of track points. Each track point $P_{im}$ is a vector of flight information including time, longitude, latitude, altitude, speed, and heading angle as follows:

$$P_{im} = \{t_m, lon_m, lat_m, al_m, sp_m, ang_m\}. \tag{3}$$

The track clustering process divides all tracks in $T$ into disjoint subsets $\{C_1, C_2, \dots, C_K\}$ according to a similarity metric. Our goal for track clustering is to derive meaningful clusters that take into account the spatial and temporal properties of the track records.

Our proposed Deep Temporal Clustering (DTC) method leverages a trainable autoencoder network to map the input to a low-dimensional representation of track point sequences, which is processed with a Temporal Clustering Layer (TCL) to group tracks into clusters. The TCL extends the idea of $k$-means by allowing soft membership to generate clusters. The integration of TCL with an autoencoder allows an end-to-end process for track clustering. To facilitate the evaluation of similarity, we introduce a novel track similarity metric, namely Track Similarity (TSim), that encodes the spatial–temporal characteristics of flight tracks into a distance metric. Our DTC method incorporates TSim into the deep temporal trajectory clustering model. Without loss of generality, we adopt Deep Temporal Denoising Autoencoder (Huang & Xu, 2021) in our description of our method. For clarity, the notations used in this paper are listed in Table 1.

**Table 1**
List of notations used in this paper.

| Notation | Definition |
|---|---|
| $A_a^{im}$ | $a$-th attribute of track points $P_{im}$ |
| $\bar{A}_a$ | Average of the $a$-th attribute of tracks |
| $\mathcal{A}$ | Autoencoder |
| $ang$ | Angle of a track point |
| $C$ | Track cluster |
| $\mathrm{Con}_{im}(\cdot)$ | Convolution output of a track point |
| $d\left(P_{im}, T_j\right)$ | Distance between a track point $P_{im}$ to track $T_j$ |
| $\hat{d}\left(P_{im}, P_{jn}\right)$ | Distance between track points $P_{im}$ and $P_{jn}$ |
| $h$ | Height of a track point |
| $L$ | Number of track points in each track |
| $lat$ | Latitude of a track point |
| $lon$ | Longitude of a track point |
| $M$ | Number of attributes of a track point |
| $\mathrm{MP}_{im}(\cdot)$ | Max pooling operation |
| $N$ | Number of tracks |
| $P_{im}$ | $m$-th track point in a track |
| $\tilde{P}_{im}$ | Preprocessed $m$-th track point in track $T_i$ |
| $\tilde{P}'_{im}$ | Encoded and decoded track point |
| $p_{ik}$ | Target probability distribution |
| $q_{ik}$ | Probability of $Z_i$ belongs to cluster $C_k$ |
| $spe$ | Speed of a track point $P_{im}$ |
| $T$ | Set of flight tracks |
| $T_i$ | $i$-th track in set $T$ |
| $\tilde{T}$ | Preprocessed track data |
| $\tilde{T}'$ | Output of the deconvolution over $Z$ |
| $t$ | Time of a track point $P_{im}$ |
| $\mathrm{TSim}\left(T_i, T_j\right)$ | Track similarity between two tracks $T_i$ and $T_j$ |
| $\omega_k$ | Center of $k$-th cluster |
| $\gamma$ | Degree of freedom of $t$-distribution |
| $Z$ | Set of encoded latent representations of flight tracks |
| $Z_i$ | $i$-th encoded latent representation of a track |

### 3.1. Track similarity metric

The track point's longitude, latitude, and altitude are the indicators of flight position in the airspace, while the difference of positions between tracks reflects the spatial distance between tracks. The angle reflects the flight movement direction and determines the next flight position. Therefore, the difference of angles between the track points is also an indicator of spatial distance between tracks. The speed indicates how fast the flight position changes, which measures the track similarity since the speed results in the difference in spatial distance between tracks.

Our proposed TSim integrates the aforementioned track properties and computes the similarity of two tracks $T_i$ and $T_j$ as follows:

$$\mathrm{TSim}\left(T_i, T_j\right) = \frac{1}{L}\sum_{m=1}^{L} d\left(P_{im}, T_j\right), \tag{4}$$

where $L$ is the number of track points in $T_i$ and $T_j$. Function $d\left(P_{im}, T_j\right)$ calculates the distance between a point $P_{im}$ in track $T_i$ to track $T_j$, which is the average distance between $P_{im}$ and every point in $T_j$:

$$d\left(P_{im}, T_j\right) = \frac{1}{L}\sum_{n=1}^{L} \hat{d}\left(P_{im}, P_{jn}\right), \tag{5}$$

where $P_{im} \in T_i$ and $P_{jn} \in T_j$.

To accommodate the different ranges of the track point properties, we compute the point-wise distance $\hat{d}$ using the normalized exponential distance as follows:

$$\hat{d}\left(P_{im}, P_{jn}\right) = \frac{1}{M}\sum_{a=1}^{M} e^{-\frac{\left|A_a^{im} - A_a^{jn}\right|}{\bar{A}_a}}, \tag{6}$$

where $A_a^{im}$ and $A_a^{jn}$ are the $a$th attributes of track points $P_{im}$ and $P_{jn}$, respectively. $\bar{A}_a$ is the average of the $a$th attribute of the tracks $T_i$ and $T_j$ and $M$ is the total number of attributes in one track point.

This track similarity metric takes the form of an exponential function with a negative first-order norm of the difference of the corresponding attributes of two track points. The negative exponential function is a monotonically decreasing function and maps the difference in attributes to interval $[0, 1]$. This similarity metric gives a large similarity value to track points that are close and a small value to track points that are far apart.

### 3.2. Deep Spatial–Temporal Clustering Network

Our proposed Deep Spatial–Temporal Clustering Network consists of two main components: an autoencoder network and a Temporal Clustering Layer (TCL). Fig. 1 illustrates the flow chart of our network including the data preprocessing step.

1. The autoencoder extracts features from the input track data with suppressed distortions, which serves as the input for the TCL component. The outputs of the encoder layers are used to compute the difference between tracks.

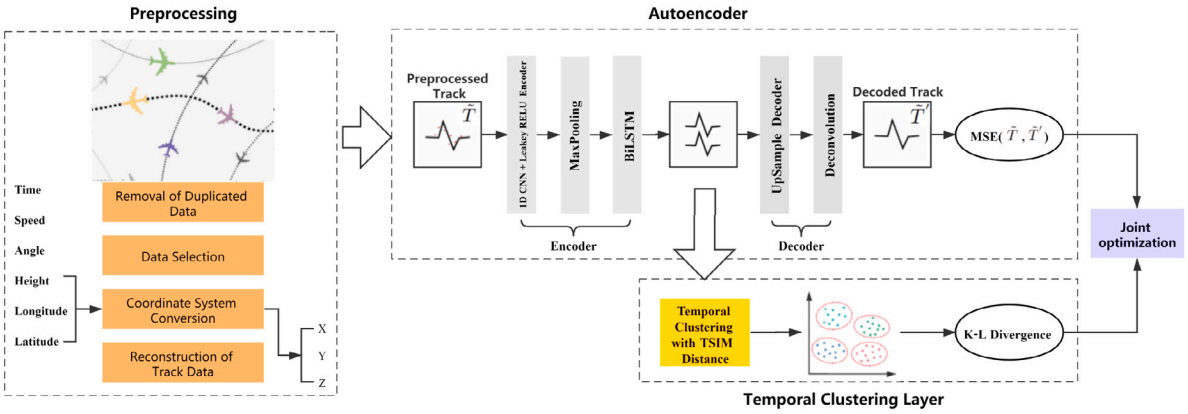


**Fig. 1.** Flow chart of our proposed Deep Spatial–Temporal Clustering method.

2. TCL takes the features from the encoder and groups the tracks into clusters using our proposed TSim metric. The output of TCL is integrated with the track difference from the autoencoder to compute the final clusters.

In this figure, the illustration of autoencoder follows Deep Temporal Denoising Autoencoder architecture, which can be replaced with a different one. The objective function for the autoencoder is based on Mean Square Error (MSE) and the objective function of the TCL component is based on Kullback–Leibler (K–L) divergence. Our method optimizes the clustering and autoencoder by minimizing the total loss of K–L divergence and MSE. We first pretrain parameters of the autoencoder to obtain the latent representation. After pretraining, we initialize the cluster centers by hierarchical clustering. We then update the autoencoder weights and cluster centers using Adam optimizer. This optimization process continues until no more than 0.1% of cluster assignment is changed.

### 3.2.1. Feature encoding

In the data preprocessing step, we clean and prepare track data for learning, which includes the operations of removal of duplicated data, data selection, coordinate system conversion, and reconstruction of track data, as shown in the preprocessing block of Fig. 1. The autoencoder network takes the preprocessed data to extract the latent representation of tracks. The output of the encoder is the latent representation $Z$ and the new track point $\tilde{P}'_{im}$ after encoding and decoding. As shown in Fig. 1, the DTDA network consists of four components: convolution layers, max-pooling layers, bi-directional LSTM networks, and decoder. The convolution layers, max-pooling layers, bi-directional LSTM networks form the encoder of the network and the decoder upsamples the inputs and performs deconvolution.

**Convolutional and Max Pooling Layers:** One-dimensional convolution operation is performed via Eq. (7) on each track point $\tilde{P}_{im} = \{\tilde{t}_{im}, \tilde{lon}_{im}, \tilde{lat}_{im}, \tilde{al}_{im}, \tilde{sp}_{im}, \tilde{ang}_{im}\}$ which is the preprocessed track data. We use full convolution in the convolutional layer. One-dimensional convolution kernel mapping can capture the latent characteristics of short-distance fluctuation between track sequences. We then perform max pooling operation with kernel size $r$ to reduce dimensions with Eq. (8). The activation function is Leaky ReLU. After all the operations above, the track sequence data are compressed into a compact vector, while keeping the structured information between sequences.

$$Con_{im}(x) = \sum_{j=0}^{\kappa-1} g(\kappa - j)\tilde{A}^{im}_{x-j} \tag{7}$$

where $\kappa$ is the kernel size of one-dimensional convolution layer and $g(\kappa - j)$ is the $(\kappa - j)$-th one-dimensional convolution kernel. $Con_{im}$ is the output sequence of $m$th point in track $\tilde{T}_i$ after one-dimensional convolution, where $Con_{im}(x)$ is the $x$th one-dimensional convolution output of track point $\tilde{P}_{im}$ $(x \in \{1, 2, \ldots, M + K - 1\})$.

$$MP_{im}(y) = \max\{Con_{im}(1 + (y-1) \times stride), Con_{im}(2 + (y-1) \times stride), \ldots, Con_{im}(r + (y-1) \times stride)\} \tag{8}$$

where $r$ is the size of pooling kernel, and $stride$ is the pooling step size. $MP_{im}$ is the output sequence of $m$th track point in track $\tilde{T}_i$ after one-dimensional max pooling, where $MP_{im}(y)$ is the $y$th one-dimensional max pooling output value of track point $\tilde{P}_{im}$ $(y \in \{1, 2, \ldots, \lfloor \frac{length[Con_{im}(x)]-r}{stride} \rfloor + 1\})$.

**Bi-directional LSTM Network:** The output of max-pooling is fed into a bi-directional LSTM network (Cai, Liu, Wei, Li, & Kan, 2021), as shown in Fig. 2. The BiLSTM network can learn the characteristics of track sequences from two directions and compress the input sequence into a more compact representation. BiLSTM is a combination of a forward LSTM and a backward LSTM. The calculation process of LSTM is summarized as follows: useful information is transmitted for subsequent calculations by forgetting and remembering new information in the cell state, meanwhile useless information is discarded. The hidden layer representation $Z$ is the required output. Convolutional layers, max-pooling layers, and bi-directional LSTM network complete the encoding process and get the latent representation $Z$ of flight tracks.

**Decoder:** The encoded latent representation $Z$ is converted to a track $\tilde{T}'$, which is of the same size as the input track $\tilde{T}$. An up-sampling operation is applied to $Z$ that changes its size from $N \times L/\text{stride} \times \text{num}_{\text{units}}$ to $N \times L \times \text{num}_{\text{units}}$, where $N$, $L$, stride, and $\text{num}_{\text{units}}$ are the number of tracks, the number of track points, the pooling step size, and the number of units in the BiLSTM components, respectively. A deconvolution with up-sampling is then performed to reconstruct the track $\tilde{T}'$. The size of tracks $\tilde{T}$ and $\tilde{T}'$ is the same and is of $N \times L \times M$, where $M$ is the number of attributes of a track point.

**Loss Function:** We use Mean Squared Error (MSE) to measure the difference between reconstructed output track $\tilde{T}'$ and input track $\tilde{T}$ after data preprocessing:

$$\frac{1}{L}\sum_{m=1}^{L}\left(\tilde{P}_{im} - \tilde{P}'_{im}\right)^2 \quad \forall i \in [1, N] \tag{9}$$

where $\tilde{P}'_{im}$ and $\tilde{P}_{im}$ are points of tracks $\tilde{T}'$ and $\tilde{T}$, respectively.

### 3.2.2. Temporal clustering layer

Our temporal clustering layer takes the latent representation $Z_i$ generated by the encoder as the input and employs a soft $k$-means algorithm to generate the clustering results. The network structure of TCL is illustrated in Fig. 3. Given $k$ cluster centers, denoted with $\omega_k$, similarity of each latent representation $Z_i$ to $\omega_k$ is computed following

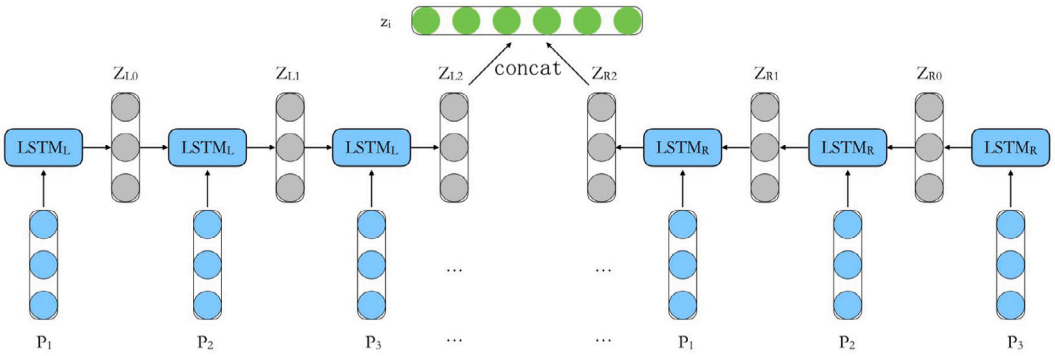


**Fig. 2.** Structure of BiLSTM.

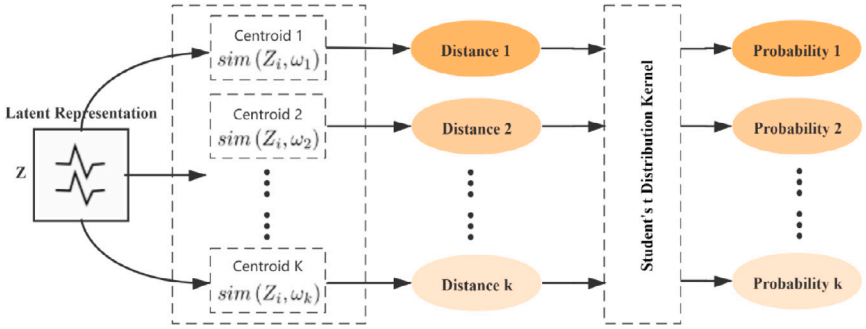


**Fig. 3.** Structure of the temporal clustering layer.

Eq. (6). A distribution kernel function is applied to modulate the distance for computing the probability of cluster association.

We adopt Student's $t$-distribution (Maaten & Hinton, 2008) to modulate the distance, which results in a probability (i.e., a soft membership) to clusters instead of the hard membership in the conventional $k$-means. The computation of the membership probability is as follows:

$$q_{ik} = \frac{\left(1 + \frac{Sim(Z_i,\omega_k)}{\gamma}\right)^{-\frac{\gamma+1}{2}}}{\sum_{k=1}^{K}\left(1 + \frac{Sim(Z_i,\omega_k)}{\gamma}\right)^{-\frac{\gamma+1}{2}}} \tag{10}$$

where $q_{ik}$ is the probability of $Z_i$ belonging to cluster $C_k$ with center $\omega_k$, and $\gamma$ is the degree of freedom of $t$-distribution. When the degree of freedom is small, the distribution function has heavy tails in comparison to the widely used normal distribution. This property diversifies the member probability to allow the association of a track to more than one cluster. In our experiments, the degree of freedom is set to one.

We calculate the loss of clustering using K–L divergence, which is an asymmetric measurement of the difference between two probability distributions. We obtain the target distribution $p$ from distribution $q$ following the method in Hinton, Osindero, and Teh (2006), Xie, Girshick, and Farhadi (2016).

$$p_{ik} = \frac{q_{ik}^2/f_k}{\sum_{k=1}^{K} q_{ik}^2/f_k} \tag{11}$$

where $f_k = \sum_{i=1}^{N} q_{ik}$.

The loss of clustering is calculated as follows:

$$\sum_{i=1}^{N}\sum_{k=1}^{K} p_{ik} log \frac{p_{ik}}{q_{ik}} \tag{12}$$

where $N$ is the total number of tracks, and $K$ is the number of clusters.

Algorithm 1 presents our Deep Spatial–Temporal Clustering Network. Note that the loop is used to perform the joint optimization process in the flow chart of Fig. 1. The convergence evaluation is based on changes to the clusters between iterations. In our experiments, the threshold of changes to the clusters is 0.1%.

**Algorithm 1** Deep Spatial–Temporal Clustering Network

**Require:** Track data, Number of clusters
**Ensure:** Track clusters $C$
1: Initialize the autoencoder $\mathcal{A}$
2: Initialize clusters $C$
3: Pretrain $\mathcal{A}$ using MSE loss Eq. (9)
4: **while** not converged or not reaching the maximum epoches **do**
5: Compute the latent representation $Z$ using the encoder of $\mathcal{A}$
6: Generate $C$ given $Z$ following Eq. (4)
7: Compute K–L divergence of $C$ following Eq. (12)
8: Compute loss of $\mathcal{A}$ following Eq. (9)
9: Update $\mathcal{A}$ following Adam optimizer
10: **end while**

## 4. Experimental results

### 4.1. Datasets and experimental settings

In our experiments, we use aircraft Automatic Dependent Surveillance-Broadcast (ADS-B) data made available by VariFlight to evaluate our proposed method and conduct a comparison study. ADS-B is a surveillance technology used by aircraft to determine its position and periodically broadcast it such that the ground control can track the aircraft's status. The system acquires the flight information at a fine temporal scale that includes flight position, speed, heading, airline, and flight numbers. The track data of a flight is a sequence of records. Table 2 presents a section of one flight track used in our experiments. Among the available track properties, UTC time, airline number (anum), and flight number (fnum) of the flight data are not used in the clustering process.

To evaluate our proposed method, we collected 800 flight tracks of 16 flights of Shanghai Hongqiao International Airport from VariFlight.

**Table 2**
Sample of flight track data. Columns anum and fnum are airline number and flight number respectively, which are not used in the similarity computation.

| Time | UTC Time | anum | fnum | Altitude | Speed | Angle | Longitude | Latitude |
|---|---|---|---|---|---|---|---|---|
| 1625107644 | 2021-07-01 02:47:24 | B8562 | MU6347 | 0.0 | 107.416 | 177 | 121.330138 | 31.212891 |
| 1625107679 | 2021-07-01 02:47:59 | B8562 | MU6347 | 152.4 | 318.544 | 177 | 121.330138 | 31.212891 |
| 1625107690 | 2021-07-01 02:48:10 | B8562 | MU6347 | 259.08 | 311.136 | 176 | 121.33306 | 31.18776 |
| 1625107693 | 2021-07-01 02:48:13 | B8562 | MU6347 | 350.52 | 311.136 | 176 | 121.33306 | 31.18776 |
| 1625107700 | 2021-07-01 02:48:20 | B8562 | MU6347 | 396.24 | 311.136 | 176 | 121.33306 | 31.18776 |
| 1625107704 | 2021-07-01 02:48:24 | B8562 | MU6347 | 480.06 | 311.136 | 176 | 121.33306 | 31.18776 |
| 1625107715 | 2021-07-01 02:48:35 | B8562 | MU6347 | 579.12 | 311.136 | 176 | 121.33446 | 31.16006 |
| 1625107719 | 2021-07-01 02:48:39 | B8562 | MU6347 | 624.84 | 311.136 | 176 | 121.33446 | 31.16006 |
| 1625107735 | 2021-07-01 02:48:55 | B8562 | MU6347 | 739.14 | 320.396 | 178 | 121.33457 | 31.15443 |
| 1625107738 | 2021-07-01 02:48:58 | B8562 | MU6347 | 754.38 | 320.396 | 178 | 121.33493 | 31.14429 |

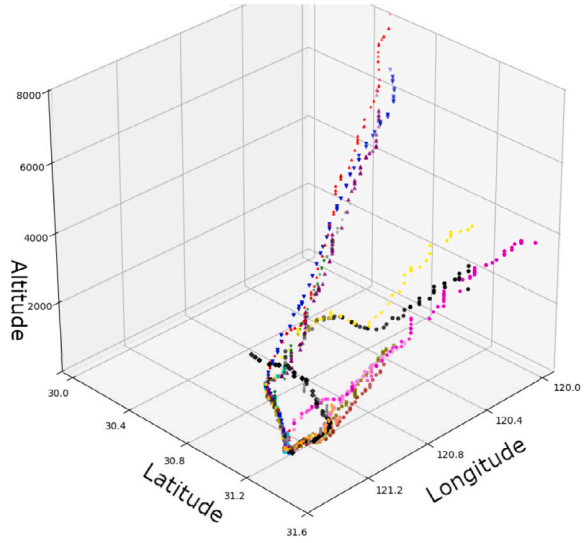


**Fig. 4.** Sample tracks of the 16 flights. To avoid clutter, one track per flight is depicted.

Each of the 16 flights has 50 tracks and each track consists of 100 records (a.k.a. points). Among the 16 flights, eight are in-bounding and eight are out-bounding flights. These flight tracks represent the flight status near the terminal area of the airport. That is, for the in-bounding flights, 100 track points at the end of the entire track near the airport are used in our experiments. For the out-bounding flights, the first 100 points close to the start of the flight tracks are used. The track segments include a few records of taxiing before take-off or after landing and most of the records of ground maneuver are excluded from our samples. Fig. 4 illustrates sample flight tracks of our experiment dataset. Points on flight tracks are spatially uneven due to the flight speed and sampling process. The tracks of take-off flights are often spatially sparse point distribution and gradually wider track coverage, while the landing flights illustrate the opposite characteristics.

In our evaluation, we repeat each experiment fifteen times and report the average results. We adopt the Silhouette Coefficient, Calinski–Harabase index, and Davies–Bouldin (DB) index as metrics of clustering performance. *Silhouette coefficient* combines cohesion and dispersion as a measurement of clustering performance and is computed as follows:

$$SC(i) = \frac{dis(i) - coh(i)}{max\{coh(i), dis(i)\}} \tag{13}$$

where $coh(i)$ is the average distance between track $T_i$ and other tracks within the same cluster, indicating intra-cluster similarity of track $T_i$. $dis(i)$ is the minimum value of the average distance between track $T_i$ and all tracks in other clusters, which computes the inter-cluster dissimilarity of $T_i$. The range of the Silhouette coefficient is $[-1, 1]$ and a large Silhouette coefficient (close to 1) indicates track $T_i$ is highly likely to be clustered to the right group; whereas a small Silhouette coefficient (close to $-1$) indicates that track $T_i$ is inappropriately clustered.

*Calinski–Harbasz index* evaluates the variance between clusters and within the cluster as a measurement of clustering performance. It is computed as follows:

$$CH = \frac{SS_B}{K-1} / \frac{SS_W}{N-K} \tag{14}$$

where $SS_B$ is the variance between clusters, and $SS_W$ is the variance within clusters. The Calinski–Harabase index is the ratio of the degree of separation and compactness. The compactness of the cluster is calculated as the sum of the square distance between every point in a cluster and the center of this cluster. The degree of separation between datasets is calculated with the sum of the square distance between the cluster centers and the center of the whole dataset. Therefore, a large Calinski–Harabase index indicates better clustering performance.

*Davies–Bouldin index* combines inter-cluster distance and intra-cluster dispersion to determine the performance of the clustering algorithm, which is computed as follows:

$$DB = \frac{1}{K}\sum_{k=1}^{K} \underset{k'=1,\ldots,K, k'\neq k}{max} \left(\frac{\sigma_k + \sigma_{k'}}{d_{kk'}}\right) \tag{15}$$

where $\sigma_k$ is the intra-cluster dispersion of $k$th cluster, and $d_{kk'}$ is the inter-cluster distance between $k$th and $k'$-th clusters. A small Davies–Bouldin index means a favorable clustering performance.

### 4.2. Analysis of track similarity and model parameters

Similarity metrics play an important role in clustering methods. In our analysis of flight tracks, our proposed similarity metric TSim integrates spatial and temporal properties to assist clustering. To understand its impact on clustering results, we conduct a comparison study again Manhattan Distance, Pearson Correlation, and Jeffrey's divergence (Zhang & Shi, 2021). In our experiments, we replace TSim of our method with each of these metrics and conduct clustering of 800 flight tracks. The average clustering performance is reported in Table 3 and the numbers in parenthesis are respective standard deviation. The best results are highlighted in bold and the second best is marked with an underscore.

Among all three performance metrics, TSim excels in terms of the Silhouette coefficient and Davies–Bouldin index and ranks the second best in terms of the Calinski–Harabase index. TSim also exhibits highly competitive consistency with a small standard deviation in all cases. Due to the increase of the speed, the track of taking-off flights shows the characteristic of the track points from dense to sparse, and the track coverage is wider. On the other hand, the track of landing flights shows the opposite characteristics. TSim explores the temporal characteristics to facilitate clustering flights. Consequently, we observe an improvement of clustering results using TSim over other similarity metrics. In comparison to Manhattan distance that performed well, the improvements in terms of the Silhouette coefficient and Davies–Bouldin index are 14.3% and 14.7%, respectively.

Manhattan distance measures the track similarity according to space distances between tracks. Jeffrey's divergence is the symmetric version of the K–L divergence, and also reflects the spatial similarity

**Table 3**
Clustering performance of using different track similarity metrics. The standard deviation is in parenthesis.

| Similarity metric | Silhouette coefficient | Calinski–Harabase index | Davies–Bouldin index |
|---|---|---|---|
| Manhattan Distance | 0.49 (0.02) | 1652.64 (141.20) | 0.78 (0.03) |
| Pearson Correlation | 0.20 (0.03) | **2083.19** (306.25) | 2.57 (1.10) |
| Jeffrey's Divergence | 0.47 (0.04) | 1473.74 (200.63) | 0.88 (0.09) |
| TSim (our) | **0.57** (0.03) | 1655.22 (133.44) | **0.67** (0.08) |

**Table 4**
Clustering performance using different convolution kernel sizes $r$.

| $r$ | Silhouette coefficient | Calinski–Harabase index | Davies–Bouldin index |
|---|---|---|---|
| 2 | 0.34 (0.07) | 540.64 (198.27) | 1.09 (0.16) |
| 3 | **0.57** (0.03) | **1655.22** (133.44) | **0.67** (0.08) |
| 4 | 0.44 (0.12) | 839.06 (446.97) | 0.88 (0.22) |
| 5 | 0.31 (0.11) | 541.58 (240.83) | 1.42 (0.63) |
| 6 | 0.37 (0.17) | 870.38 (799.61) | 1.31 (0.55) |

between flights. In the space close to the airport, the flight tracks are regulated by ground air traffic control. Therefore, both Manhattan distance and Jeffrey's divergence achieved good performance in terms of the Silhouette coefficient and Davies–Bouldin index. However, the uneven spatial distance among track points makes it difficult for the two metrics to capture the temporal differences. Pearson correlation measures the cooccurrence of patterns among random variables, which is less dependent on the magnitude of the track properties. It resulted in the largest Calinski–Harabase index among all metrics but performed poorly in terms of the Silhouette coefficient and Davies–Bouldin index. TSim ranks the second best for the Calinski–Harabase index metric.

A key parameter of our proposed method is the size of the convolution kernels. Our DTC method has 50 convolution kernels and the pooling size is two to make the size of latent representation less than 100 for the sake of computational efficiency. The size of the deconvolution kernels of the decoder is the same as that of the convolution kernels. To understand the impact of kernel size of the one-dimensional convolutional layer on the performance of DTC, we conducted experiments using different kernel sizes ranging from 2 to 6.

Table 4 presents the average clustering performance of our DTC method using different kernel sizes. Again, the standard deviation is in parentheses. The best results are highlighted in bold and the second best is marked with an underscore. There is no clear trend of performance change with respect to the kernel size. In general, better performance is achieved when the kernel size is about 3 and 4. However, size three stands out with a much consistently better performance in terms of both average performance and standard deviation. With respect to the three metrics, it demonstrates superior results to the others by at least 27% in comparison to the second best. Hence, in the rest of our experiments, we set the kernel size to three in both convolution and deconvolution processes for our DTC method.

### 4.3. Comparison study

We compare the performance of our proposed DTC method against the classical and state-of-the-art methods including $k$-means, $k$-medoids, Relation-Guided Representation Learning (RGRL) (Kang et al., 2020) and a variant of RGRL by integrating our TCL component. The similarity metrics used in $k$-means and $k$-medoids is TSim. Table 5 reports the average performance, together with standard deviations, of the six compared clustering methods.

All methods exhibit a positive average Silhouette coefficient, which indicates plausible clustering results. $k$-means and $k$-medoids differ in the ways of computing clustering centers. $k$-means updates the centers by the average of tracks, whereas $k$-medoids updates a center by selecting the track closest to the previous center. The better performance by $k$-means clustering in contrast to $k$-medoids can probably be attributed to the property of the datasets, that is, dense flight tracks near the airport terminal area. The interpretation of the performance of RGRL is arguable. It exhibits inferior performance in terms of the Silhouette coefficient and Davies–Bouldin index but results in much better performance (an improvement of 41.9%) in terms of the Calinski–Harabase index. In comparison to $k$-means, $k$-medoids, and RGRL method, DTC achieves the largest Silhouette coefficient, that is, it improves the performance by more than 62% in comparison to the second best. The performance improvements are also significant when the Calinski–Harabase index and Davies–Bouldin index are used as metrics with an improvement of 95.6% and 57.3%, respectively.

RGRL leverages sample relations to guide the representation learning to preserve the local neighborhood structure on the data manifold and maintains its consistency in both input and embedding spaces. We conduct experiments by integrating our TCL component with the latent representation extracted by RGRL. The row of RGRL+TCL in Table 5 presents the results of integration. In contrast to the RGRL method, the integration of RGRL and TCL suffers performance degradation. This suggests that the choice of feature embedding strategy is critical for our method. The Deep Temporal Denoising AutoEncoder provides suitable latent representations of the flight tracks for clustering.

In our evaluation, we want to understand how the temporal components affect the clustering performance of our method. We conduct experiments by removing the time property from our track records and repeating the experiments using the exact same parameters of our DTC method. That is, we eliminate the influence of temporal characteristics to track clustering. The row of DTC (w/o time) in Table 5 presents our results. It is clear that the average performance of DTC degrades when temporal data are absent. The amount of degradation is substantial in terms of the Silhouette coefficient and Calinski–Harabase index at 24.6% and 37.7%, respectively.

### 4.4. Robustness to noise

We also analyze the performance of our method in the presence of noise distortion. Noise is added to each component of the track record and follows Gaussian distribution with zero mean and a variance in the range of $[0.1, 0.5]$. Table 6 lists the average performance of our proposed DTC method. In the first row where the noise variance is zero, the results are produced by processing the original track records without additive noise. There is a clear trend that as we increase noise variance, the performance of DTC reduces. Both the Silhouette coefficient and Calinski–Harabase index reduce and the Davies–Bouldin index increases without an exception. When we look at the performance between two adjacent noise variances, we observe a relatively large drop of performance between no noise and noise with a variance of 0.1. The change is more than 10%. As noise continues increasing, the performance degradation slows down. The change is less than 7% and, in some cases, is close to zero. This demonstrates the robustness of our method with respect to noise.

In contrast to the clustering performance by the state-of-the-art methods reported in Table 5, the performance of DTC is still very

**Table 5**
Clustering performance of classical and state-of-the-art methods.

| Method | Silhouette coefficient | Calinski–Harabase index | Davies–Bouldin index |
|---|---|---|---|
| $k$-means | 0.35 (0.03) | 596.61 (37.29) | 1.57 (0.09) |
| $k$-medoids | 0.26 (0.09) | 446.79 (26.28) | 2.19 (0.35) |
| RGRL | 0.30 (0.00) | 846.75 (5.46) | 3.57 (0.57) |
| RGRL+TCL | 0.21 (0.00) | 453.49 (8.98) | 27.16 (10.08) |
| DTC (w/o time) | 0.43 (0.17) | 1031.16 (793.71) | 0.70 (0.13) |
| DTC | **0.57** (0.03) | **1655.22** (133.44) | **0.67** (0.08) |

**Table 6**
Clustering performance of DTC under Gaussian noise of different degrees.

| Variance | Silhouette coefficient | Calinski–Harabase index | Davies–Bouldin index |
|---|---|---|---|
| 0 | 0.56 | 1667.78 | 0.68 |
| 0.1 | 0.50 | 1045.84 | 0.85 |
| 0.2 | 0.48 | 991.45 | 0.90 |
| 0.3 | 0.46 | 835.69 | 0.93 |
| 0.4 | 0.45 | 834.34 | 0.95 |
| 0.5 | 0.42 | 774.06 | 0.95 |

competitive even under noise distortion. When there is no noise, the methods that achieve the second best performance are $k$-means (in terms of Silhouette coefficient at 0.35), RGRL (in terms of Calinski–Harabase index at 843.32), and $k$-means (in terms of Davies–Bouldin index at 1.57). Our proposed method achieves favorable performance even under the impact of noise with as much as 0.4 in variance. It confirms that our method is reasonably robust to noise.

## 5. Conclusion

This paper introduces a track similarity based on the spatial–temporal characteristics of flight tracks and presents a Deep Temporal Clustering method using a denoising autoencoder. Our proposed method employs the Deep Temporal Denoising Auto-encoding network to extract the latent representations of the track sequences. By extending the idea of $k$-means clustering, DTC clusters the flight tracks with a Time Clustering Layer.

Experiments are conducted using ADS-B track data. We analyze the effectiveness of the proposed track similarity metric TSim in the framework of our proposed method and compared it against Manhattan distance, Pearson correlation, and Jeffrey's divergence. It is demonstrated that TSim excels in terms of the Silhouette coefficient and Davies–Bouldin and exhibits highly competitive consistency with a small standard deviation. In the evaluation of the clustering performance, our comparison study includes classical methods as well as state-of-the-art methods. Among all cases, DTC achieved much-improved performance in terms of three metrics by more than 57.3% in contrast to the second best. By removing the temporal component, we observe clear degradation of the clustering performance. When we introduce noise to the track records, the performance of our method drops. However, as the noise continues increasing, the performance degradation slows down. The change is less than 7% and, in some cases, is close to zero, which demonstrates the robustness of our method with respect to noise. It is worth mentioning that the performance of DTC when processing tracks with the noise of variance at 0.4 is superior to the compared methods that cluster tracks without noise distortion.

## CRediT authorship contribution statement

**Guoqian Liu:** Software, Writing – original draft, Writing – review & editing. **Yuqi Fan:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing. **Jianjun Zhang:** Project administration, Resources. **Pengfei Wen:** Validation, Resources. **Zengwei Lyu:** Validation, Resources. **Xiaohui Yuan:** Conceptualization, Formal analysis, Writing – original draft, Writing – review & editing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work was partly supported by the National Natural Science Foundation of China (62002097), the Key Research and Development Program of Anhui Province, China (201904a07020030), the open project of State Key Laboratory of Complex Electromagnetic Environment Effects on Electronics and Information System in China (CEMEE2018Z0102B), and the open fund of Intelligent Interconnected Systems Laboratory of Anhui Province, China (PA2021AKSK0114), Hefei University of Technology.

## References

Ankerst, M., Breunig, M. M., Kriegel, H.-P., & Sander, J. (1999). OPTICS: Ordering points to identify the clustering structure. In *ACM SIGMOD international conference on management of data, Vol.28* (2), (pp. 49–60). ACM.

Besse, P. C., Guillouet, B., Loubes, J.-M., & Royer, F. (2016). Review and perspective for distance-based clustering of vehicle trajectories. *IEEE Transactions on Intelligent Transportation Systems*, *17*(11), 3306–3317.

Cai, G., Lee, K., & Lee, I. (2018). Mining mobility patterns from geotagged photos through semantic trajectory clustering. *Cybernetics and Systems*, *49*(4), 234–256.

Cai, W., Liu, B., Wei, Z., Li, M., & Kan, J. (2021). TARDB-Net: triple-attention guided residual dense and BiLSTM networks for hyperspectral image classification. *Multimedia Tools and Applications*, *80*(7), 11291–11312.

Campello, R. J., Moulavi, D., & Sander, J. (2013). Density-based clustering based on hierarchical density estimates. In *Pacific-asia conference on knowledge discovery and data mining, Vol. 7819* (pp. 160–172). Springer-Verlag.

Conde Rocha Murca, M., DeLaura, R., Hansman, R. J., Jordan, R., Reynolds, T., & Balakrishnan, H. (2016). Trajectory clustering and classification for characterization of air traffic flows. In *16th AIAA aviation technology, integration, and operations conference* (p. 3760).

Dahlbom, A., & Niklasson, L. (2007). Trajectory clustering for coastal surveillance. In *2007 10th international conference on information fusion* (pp. 1–8). IEEE.

Devarakonda, N., Saidala, R. K., & Kamarajugadda, R. (2021). A hybrid between TOA and Lévy flight trajectory for solving different cluster problems. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, *15*(4), 1–25.

Eckstein, A. (2009). Automated flight track taxonomy for measuring benefits from performance based navigation. In *2009 integrated communications, navigation and surveillance conference* (pp. 1–12). IEEE.

Enriquez, M. (2013). Identifying temporally persistent flows in the terminal airspace via spectral clustering. In *10th USA/Europe air traffic management research and development seminar (ATM2013)/federal aviation administration (FAA) and EUROCONTROL* (pp. 10–13).

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD, Vol.96* (34), (pp. 226–231).

Gariel, M., Srivastava, A. N., & Feron, E. (2011). Trajectory clustering and an application to airspace monitoring. *IEEE Transactions on Intelligent Transportation Systems*, *12*(4), 1511–1524.

Golay, X., Kollias, S., Stoll, G., Meier, D., Valavanis, A., & Boesiger, P. (1998). A new correlation-based fuzzy logic clustering algorithm for FMRI. *Magnetic Resonance in Medicine*, *40*(2), 249–260.

Gui, X., Zhang, J., & Peng, Z. (2021). Trajectory clustering for arrival aircraft via new trajectory representation. *Journal of Systems Engineering and Electronics*, *32*(2), 473–486.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.

Huang, Y., & Xu, Q. (2021). Electricity theft detection based on stacked sparse denoising autoencoder. *International Journal of Electrical Power & Energy Systems*, *125*, Article 106448.

Kang, Z., Lu, X., Liang, J., Bai, K., & Xu, Z. (2020). Relation-guided representation learning. *Neural Networks*, *131*, 93–102.

Krishnapuram, R., Joshi, A., & Yi, L. (1999). A fuzzy relative of the k-medoids algorithm with application to web document and snippet clustering. In *FUZZ-IEEE'99. 1999 IEEE international fuzzy systems. conference proceedings (Cat. No. 99CH36315), Vol. 3* (pp. 1281–1286). IEEE.

Lee, J.-G., Han, J., & Li, X. (2008). Trajectory outlier detection: A partition-and-detect framework. In *2008 IEEE 24th international conference on data engineering* (pp. 140–149). IEEE.

Li, H. (2021). Time works well: Dynamic time warping based on time weighting for time series data mining. *Information Sciences*, *547*, 592–608.

Lin, Z., Kang, Z., Zhang, L., & Tian, L. (2021). Multi-view attributed graph clustering. *IEEE Transactions on Knowledge and Data Engineering ( Early Access )*, 1.

Liu, C., & Guo, C. (2020). STCCD: Semantic trajectory clustering based on community detection in networks. *Expert Systems with Applications*, *162*, Article 113689.

Liu, Y., Liu, J., Jin, Y., Li, F., & Zheng, T. (2020). An affinity propagation clustering based particle swarm optimizer for dynamic optimization. *Knowledge-Based Systems*, *195*, Article 105711.

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, *28*(2), 129–137.

Lv, J., Kang, Z., Lu, X., & Xu, Z. (2021). Pseudo-supervised deep subspace clustering. *IEEE Transactions on Image Processing*, *7*, 184985–185000.

Maaten, L. v. d., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(11), 2579–2605.

Murça, M. C. R., Hansman, R. J., Li, L., & Ren, P. (2018). 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)*, *97*, 324–347.

Niedermeier, R., & Sanders, P. (1996). *On the manhattan distance between points on space filling mesh indexings*. Citeseer.

Olive, X., & Morio, J. (2019). Trajectory clustering of air traffic flows around airports. *Aerospace Science and Technology*, *84*, 776–781.

Oliver, X., Basora, L., Viry, B., & Alligier, R. (2020). Deep trajectory clustering with autoencoders. In *ICRAT 2020*.

Oxenham, M. (2000). Automatic air target to airlane association. In *Proceedings of the 3rd international conference on information fusion, Vol. 2* (pp. WeD4–19–WeD4–26). IEEE.

Pan, X., He, Y., Wang, H., Xiong, W., & Peng, X. (2016). Mining regular behaviors based on multidimensional trajectories. *Expert Systems with Applications*, *66*, 106–113.

Piciarelli, C., Foresti, G. L., & Snidaro, L. (2005). Trajectory clustering and its applications for video surveillance. In *IEEE conference on advanced video and signal based surveillance* (pp. 40–45). IEEE.

Qu, X., Yang, L., Guo, K., Ma, L., Sun, M., Ke, M., et al. (2021). A survey on the development of self-organizing maps for unsupervised intrusion detection. *Mobile Networks and Applications*, *26*(2), 808–829.

Rehm, F. (2010). Clustering of flight tracks. In *AIAA infotech@aerospace 2010* (p. 3412).

Seal, A., Karlekar, A., Krejcar, O., & Gonzalo-Martin, C. (2020). Fuzzy c-means clustering using Jeffreys-divergence based similarity measure. *Applied Soft Computing*, *88*, Article 106016.

Sun, M., & Wang, J. (2021). An approach of ship trajectory clustering based on minimum bounding rectangle and buffer similarity. In *IOP conference series: earth and environmental science, Vol. 769*. (3), IOP Publishing, Article 032017.

Varlamis, I., Sardianos, C., Bogorny, V., Alvares, L. O., Carvalho, J. T., Renso, C., et al. (2021). A novel similarity measure for multiple aspect trajectory clustering. In *Proceedings of the 36th annual ACM symposium on applied computing* (pp. 551–558).

Wang, L., Chen, P., Chen, L., & Mou, J. (2021). Ship AIS trajectory clustering: An HDBSCAN-based approach. *Journal of Marine Science and Engineering*, *9*(6), 566.

Xie, J., Girshick, R., & Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In M. F. Balcan, & K. Q. Weinberger (Eds.), *Proceedings of machine learning research: vol. 48*, *Proceedings of the 33rd international conference on machine learning* (pp. 478–487). New York, New York, USA: PMLR.

Yan, X., Razeghi-Jahromi, M., Homaifar, A., Erol, B. A., Girma, A., & Tunstel, E. (2019). A novel streaming data clustering algorithm based on fitness proportionate sharing. *IEEE Access*, *7*, 184985–185000.

Yang, Z., Tang, R., Chen, Y., & Wang, B. (2021). Spatial-temporal clustering and optimization of aircraft descent and approach trajectories. *International Journal of Aeronautical and Space Sciences*, 1–12.

Yu, Q., Luo, Y., Chen, C., & Chen, S. (2019). Trajectory similarity clustering based on multi-feature distance measurement. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, *49*(6), 2315–2338.

Zhang, T., Ramakrishnan, R., & Livny, M. (1996). BIRCH: An efficient data clustering method for very large databases. *ACM SIGMOD Record*, *25*(2), 103–114.

Zhang, Y.-q., & Shi, G.-y. (2021). Trajectory similarity measure design for ship trajectory clustering. In *2021 IEEE 6th international conference on big data analytics* (pp. 181–187). IEEE.

Zhong, H., Liu, H., & Qi, G. (2021). Analysis of terminal area airspace operation status based on trajectory characteristic point clustering. *IEEE Access*, *9*, 16642–16648.

Zobel, J., & Moffat, A. (1998). Exploring the similarity space. In *Acm sigir forum, Vol. 32* (1), (pp. 18–34). NY, USA: ACM New York.