A fault mode identification methodology based on self-organizing map

Sébastien Schwartz, Juan José Montero-Jiménez, Michel Salaün, Rob A. Vingerhoeds

A fault mode identification methodology based on self-organizing map

Sébastien Schwartz[1,2] · Juan José Montero Jimenez[2,3] · Michel Salaün[2] · Rob Vingerhoeds[2]

Abstract
One of the main goals of predictive maintenance is to be able to trigger the right maintenance actions at the right moment in time building upon the monitoring of the health status of the concerned systems and their components. As such, it allows identifying incipient faults and forecasting the moment of failure at the earliest stage. Many different data-driven methods are used in such approaches (Naderi and Khorasani in 2017 IEEE 30th Canadian conference on electrical and computer engineering (CCECE), Windsor, ON, IEEE, pp 1–6, 2017. https://doi.org/10.1109/ccece.2017.7946715; Sarkar et al. in J Eng Gas Turbines Power 1338(8):081602, 2011. https://doi.org/10.1115/1.4002877; Svärd et al. in Mech Syst Signal Process 45(1):170–192, 2014. https://doi.org/10.1016/j.ymssp.2013.11.002; Pourbabaee et al. Mech Syst Signal Process 76–77:136–156, 2016. https://doi.org/10.1016/j.ymssp.2016.02.023). This work uses the self-organizing maps (SOMs) or Kohonen map, thanks to its ability to emphasize underlying behavior such as fault modes. An automatic fault mode detection is presented based on a SOM network and the kernel density estimation with as less as possible prior knowledge. The different SOM development steps are presented and the suitable solutions proposed to structure the approach are accompanied by mathematical methods. The generated maps are then used with kernel density analysis to isolate fault modes on them. Finally, a methodology is presented to identify the different fault modes. The work is illustrated with an aircraft jet engines case study.

Keywords Diagnostic · Fault identification · Predictive maintenance · Self-organizing map

1 Introduction

Maintenance departments are confronted with three types of maintenance: corrective maintenance (i.e., correcting systems that break down or have a deteriorated functional behavior), preventive maintenance (i.e., maintenance actions at regular intervals, to avoid break down or deterioration) and predictive maintenance (i.e., performing specific maintenance actions based on indications derived from fine analysis on data, crew reports, etc.). Predictive maintenance has seen a huge rise over the last years, essentially due to the application of neural networks to identify incipient faults and to forecast the moment of failure through the diagnostic phase. Depending on the monitored data, there are different types of classification for diagnostic system (Fig. 1). Machine fault diagnostic approaches are grouped into model-based [1–3] and data-driven [4–7] techniques.

In this paper, a hybrid approach for a "process history-based" diagnosis with quantitative data through the combination of a neural network (NN) with a probability density function (PDF) is scoped. In particular, an unsupervised neural network (UNN) type, the self-organizing map (SOM), or Kohonen map, is used. This NN has already shown its effectiveness in the past [9] but requires substantial knowledge on the neural network type itself, making it complex to comprehend and frequently requiring "manual" rework. Therefore, the presented novel approach

✉ Sébastien Schwartz
sebastien.schwartz@isae-supaero.fr

[1] SOGETI High Tech, R&D Dpt., Aeropark, 3 Chemin de Laporte, 31100 Toulouse, France

[2] ISAE-SUPAERO, Université de Toulouse, 10 Avenue Edouard Belin, 31400 Toulouse, France

[3] Tecnológico de Costa Rica Institute of Technology, Calle 15, Avenida 14., 1 km Sur de la Basílica de los Ángeles, Provincia de Cartago, Cartago 30101, Costa Rica

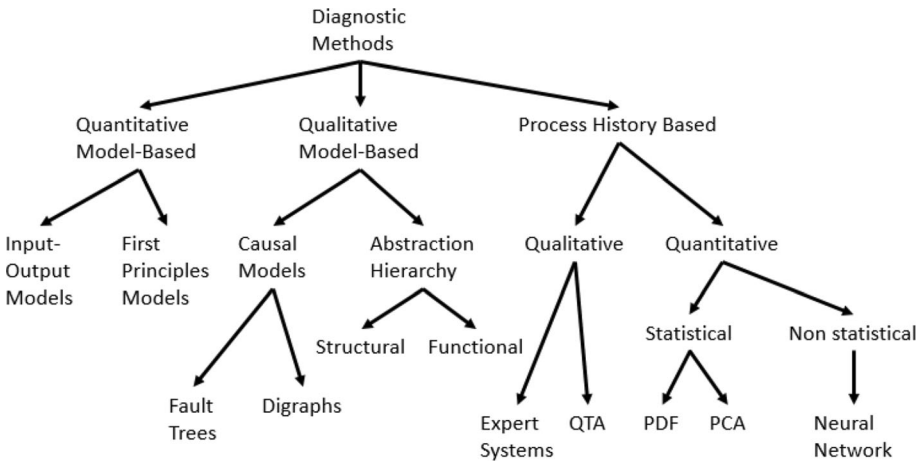


Fig. 1 Classification of diagnostic methods based on [8]. *QTA* qualitative trend analysis, *PDF* probability density function, *PCA* principal component analysis

aims at automating as much as possible for the development of the SOM. The first objective is to reduce to the minimum level of the interaction between the expert knowledge and the network itself. The complete process is reviewed, and, for each step in the process, mathematical methods are proposed. It leads to a more structured and automatable procedure with an intent to make the method more accessible and autonomous. The second objective is to manage automatically the output of the SOM with PDFs to identify fault modes.

The goal of this paper is to present a fully autonomous toolchain that identifies the fault modes from input sensors by reducing as much as possible the prior knowledge and the expert intervention.

The paper is organized as follows: In the second section, predictive maintenance is introduced. In the third section, the self-organizing map is presented, as well as the different possibilities to enhance the approach and a methodology to identify the fault modes. The fourth section exposes and applies the methodology on the case study of aircraft engines. The paper concludes with some general observations and indications for future work.

2 Predictive maintenance

Keeping a technical system in optimal operational conditions is key for a successful and an efficient use of the system. Interruptions of operations not only have a negative impact (e.g., delayed flights, internet connections not being available, etc.), but may also have worst consequences (e.g., image of the company impacted, people will tend to privilege other suppliers, loss of income, etc.). Maintaining a system in optimal state of operation also means having timely maintenance actions to ensure the intended functionality and to avoid potential failures to occur. A good combination of corrective, preventive and predictive maintenance is required [10]. Whereas corrective maintenance is based on alarm handling, troubleshooting for corrective actions, etc., predictive maintenance relies on offline diagnostic task analyses such as recorded data, crew reports, maintenance logs and other data recordings on the actual health state of the system. Such information is then used by operation departments to assess the health state and derive necessary maintenance actions and their planning if necessary.

Condition monitoring is used to assess the current health state of the system at hand. It uses pattern recognition in time series of monitored data and classifies those patterns as known conditions. While this is used to be done by human experts [9], requiring great skill and experience from the expert, software tools have appeared to support engineers in such activities.

As predictive maintenance aims to define the best possible moment to trigger maintenance actions [9], it gained more and more attention over the last few years. One of the solutions discussed in the literature for early detection and classification of failures during the diagnostic phase is the self-organizing map (SOM) originally proposed by Kohonen [11]. This approach allows to detect degradation patterns and the nature of the problem and to derive the remaining useful life [9, 10, 12]. This network has been widely used on various application fields for its particular abilities [13–18].

Visualization helps humans to understand diagnostic tasks. According to [19], the visualization of the data helps to gain an understanding of an unknown dataset. For limited amounts of dimensions in data, humans can do it, but the perception is limited to three dimensions. High-dimensional data visualization with more than three features is therefore unreachable. To address this issue, several visualization techniques were developed such as principal component analysis (PCA) [20], self-organized maps (SOMs) [11] or Sammon mapping [21]. Those techniques

rely on dimension reduction to visualize on 2D or 3D plots. This dimension reduction therefore generates new knowledge to be labeled for the analysis. Labeling data using prior knowledge or human reasoning become complicated on high dimensions [22], which could induce errors. Approaches relying on unsupervised learning are interesting thanks to their abilities to deal with high-dimensional data without prior knowledge. Therefore, a SOM network answers to the requirements: visualize high-dimensional data on 2D maps and obtain knowledge generated from these maps.

The successful implementation of SOM for diagnostic tasks requires an in-depth analysis of data obtained from the system at hand. It involves the assessment of data interdependency, the analysis on how many different faults/failures can be identified in the data, the distinction of eventual operational modes (if necessary) and, finally, successful training and subsequent validation of the SOM. Such analysis requires a good knowledge on the application domain itself and the measured data, in addition to strong knowledge on SOMs. In the next section, the fault mode diagnosis approach is presented. Each step of the SOM neural network is revisited and analyzed to see "whether and how" improvements may be obtained to automatize the use of SOM neural networks and to reduce the need for prior knowledge. Then, probability density function on the neural network output is used to identify faults of the supervised system.

3 Fault mode diagnosis using self-organizing maps

3.1 Overview

As presented previously, fault diagnosis requires human intervention and prior knowledge. The proposed methodology (Fig. 2) attempts to perform an automatic fault mode diagnosis with as less as possible prior knowledge. The self-organizing map neural network is the core of the methodology as a tool to emphasize the input data through a map representation. The underlying information such as faults becomes more accessible. The approach has been structured into three phases.

The first step is "input data management". Input data (raw monitored sensors) have to be formatted and used with the neural network. This sensor management involves the choice of useful variables to decrease the complexity and the size the neural network. In addition, the data are normalized to facilitate the network training.

The second step is "system map." The formatted input data are presented to the neural network. For the training phase, the network generates maps representing the input data. During the testing phase, the network outputs a localization on the previously trained map.

The last step is the "fault mode identification". The localization on the map (output of the previous stage) enables the identification of the fault mode thanks to a mathematical procedure based on the probability density function.

In the following sections, each step (Fig. 2) will be described more in detail.

3.2 Input data management

As presented previously, a selection among raw monitored data is performed. This procedure is called feature selection (see [23–26]). It is used on structured data to select features that explain most of the system behaviors by eliminating inappropriate and redundant data [23]. This paper will only focus on time-series data. Some basic approaches provide good means to address the feature selection. For example, the variance is a good way to eliminate features with little or no evolution. The correlation coefficient is powerful to identify feature that have the same behavior. Visual analysis highlights features that have unusual trend.

For better analysis purpose, a normalization of the input data is performed. It provides a common scale for the features. Two main methods are used: rescaling and standardizing. The rescaling method is the simplest one and consists to scale data on the range $[0, 1]$ with the following formula:

$$\bar{x}_{ij} = \frac{x_{ij} - \min_i x_{ij}}{\max_i x_{ij} - \min_i x_{ij}} \tag{1}$$

where x_{ij} and $\bar{x}_{ij}$ are, respectively, the original and normalized data values for a sample j of a feature i from the input dataset, with $i = 1...p$, where p is the number of network input.

The standardizing method uses the following formula:

$$\bar{x}_{ij} = \frac{x_{ij} - \mu_i}{\sigma_i} \tag{2}$$

where μ_i and σ_i are, respectively, the mean and the standard deviation of a feature i. Even if this method provides a uniform scale, normalized inputs data do not belong to the same common scale. That is why the rescaling method will be used to have all input data on the same range $[0, 1]$.

3.3 System map

3.3.1 Overview

Self-organizing maps are neural networks using unsupervised learning inspired from human brain way [11]. They

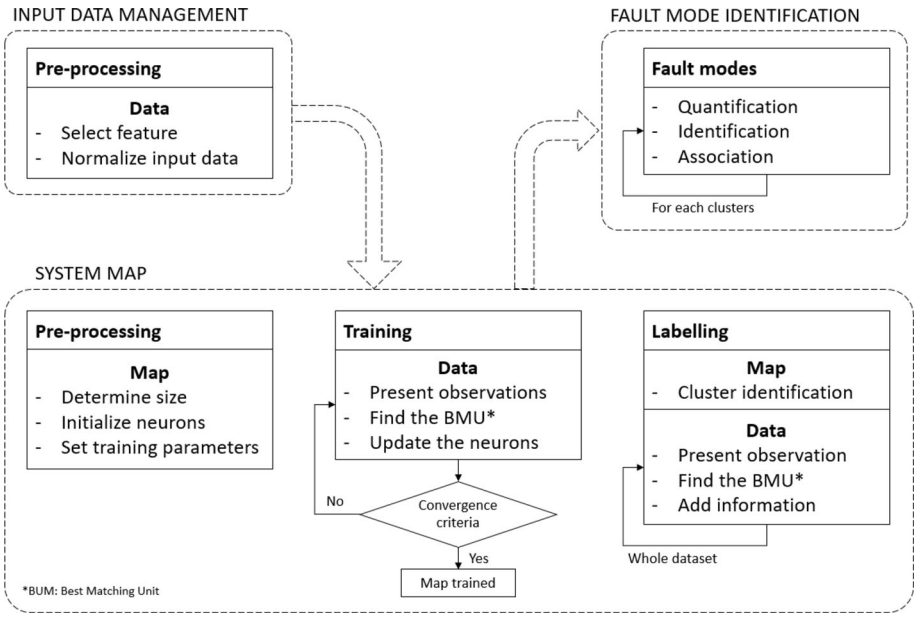


Fig. 2 Fault mode diagnosis methodology

are suitable to produce a low-dimensional representation of the input space of training samples, called a map, to visualize high-dimensional data [27]. SOMs are therefore useful for dimensionality reduction and representation in which the similarity relations between input data are preserved [10]. Its competitive learning capability and the use of the neighborhood function preserve the topological properties of the inputs. The competitive approach aims to put output neurons in competition with each other to be activated, and the winning neuron is the only one that can be activated. As most artificial neural networks, SOMs are developed in two subsequent phases: a training phase and a testing phase. Thanks to the characteristics of this particular neural network, the testing phase can also be used as a labeling part, which is the assignment of information to specific clusters on the map, such as specific faults, or system operational conditions. The goal of the training phase is to teach the algorithm with the dataset in such a way that similar data features are clustered on specific topological regions on the map [28]. Then, the generated map provides clusters, and a health index (HI) is estimated for each node, depicting the degradation status of the studied system.

The SOM neural network building is divided into three stages: preprocessing, training and labeling. A lot of manual work done by experts is needed to perform these tasks.

3.3.2 Preprocessing

SOM topologies can be in one, two or even three dimensions [29–33]. The neurons are localized at lattice nodes. The original SOM [11] is a 2D hexagonal map. Then, successively, 1D lines, 2D rectangular grids or more complex structures, such a star lattices [34] (Fig. 3), have been created. For our case study, a 2D square lattice is used to visualize it as picture.

A square lattice has $n \times n$ neurons of m weights. The number of weights per group (i.e., m) corresponds to the number of inputs to the network. According to [35], a size of the map can be determined by calculating the number of neurons from the number of observations in the dataset such as:

$$M \cong 5\sqrt{N} \tag{3}$$

where M is the number of neurons, and N is the number of observations. A square lattice will have $n = \sqrt{M}$. For example, with about 10,000 observations, Eq. (3) leads to a size $M \cong 500$. For a square lattice, the closest dimension would be a 23×23 matrix.

The next step is the map initialization. There are various ways to set initial weights, such as input vectors randomly selected [36], principal components of the input space [36], large hypercube [37] or random values. A uniform distribution in the range $[0, 1]$ with a probability density function of 1 is considered for usefulness to set neuron weight vectors $w_{ij} = \left(w_{ij1}, w_{ij2}, \ldots, w_{ijm}\right)$ with $i,j = 1\ldots n$.

3.3.3 Training phase

After the input data are processed and the map is initialized, the map is trained using preprocessed data. Training algorithms related to SOMs are various. Stochastic training as Algorithm 1 is one of the most classical algorithms [38]. Alternatives such as fast batch SOM [39] or growing self-

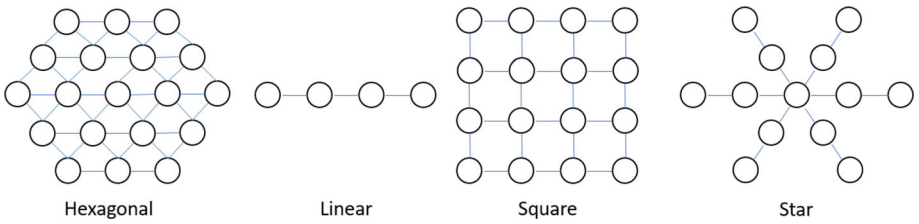


Fig. 3 Examples of lattice structures

organizing map (GSOM) [40] can be faster but are more complex to use. They all rely on the determination of the best matching unit (BMU), which is the smallest distance between the input vector and the weight vector of the map nodes.

The learning process is iterative, until a stopping criterion is met. Examples of criteria are an error estimation such as the quantization error [36] or the so-called "rule of thumb" where the number of steps must be at least 500 times the number of neurons in the map [38]. This last criterion will be used in this paper.

Algorithm 1. Randomly input selection for self- organizing map

```
For t from 1 to Number_Of_Iteration do
  | Pick up a random input vector among the input dataset
  | For each node in the map do
  |   | Calculate the distance between input vector and map node weight vector
  |   | Track the node with the smallest distance (i.e. BMU)
  | End
  | Update the weight vector of the neighborhood of the BMU node and itself
End
```

The weight vector is updated at each iteration as follows:

$$\begin{cases} w_{ij}(t+1) = w_{ij}(t) + h_{kl,ij}(t)\left[x(t) - w_{ij}(t)\right] & \forall n_{ij} \in \mathbb{E}_{BMU} \\ w_{ij}(t+1) = w_{ij}(t) & \forall n_{ij} \notin \mathbb{E}_{BMU} \end{cases} \tag{4}$$

where w_{ij} is the vector weight, t is the t^{th} iteration, $h_{kl,ij}$ is the neighborhood function, $x(t)$ is the input observed, n_{ij} is the node on the map, ij are the node coordinates on a 2D lattice, kl are the BMU node coordinates on the same 2D lattice and $\mathbb{E}_{BMU}$ is the space of BMU neighborhood node and itself. This space is defined by the width of the neighborhood function, also called the BMU radius. A smooth Gaussian kernel is mostly used for the neighborhood function [36, 41]:

$$h_{kl,ij}(t) = \eta(t) \cdot e^{\frac{-w_{kl}(t) - w_{ij}(t)_2^2}{2\sigma^2(t)}} \tag{5}$$

where $\eta(t)$ and $\sigma(t)$ are, respectively, the learning rate and the width of the kernel, which are the decreasing functions of time [36, 38]. This function decreases through the time to improve the neighborhood identification.

The BMU node n_{kl} is defined by:

$$\left\{ n_{kl} | x(t) - w_{kl2}^2 = \arg\min_{ij} x(t) - w_{ij}(t)_2^2 \right\} \tag{6}$$

The SOM training speed is linked to the map size, the number of inputs and the number of samplings. The number of weights could be large, which leads to a slow convergence due to the amount of weight updates involved in each iteration. Several mechanisms have been developed to address this problem, such as optimizing the width of the neighborhood function or learning rate function. According to [36, 41], the Gaussian kernel is a good candidate. The width of the neighborhood function $\sigma(t)$ is chosen as:

$$\sigma(t) = \sigma_0 \cdot e^{-t/\tau_1} \tag{7}$$

where σ_0 is an initial variance set to the map size divided by two [38], and τ_1 is a positive constant. The learning rate function $\eta(t)$ is chosen as:

$$\eta(t) = \eta_0 \cdot e^{-t/\tau_2} \tag{8}$$

where η_0 is an initial learning rate set to 0.9 [38], and τ_2 is a positive constant. The function is limited to a minimum set at 0.01 [38].

For convenience, τ_1 and τ_2 are equal and follow the relation:

$$\tau_i = t_{\max} / \ln \sigma_0 \quad \text{with } i = 1, 2 \tag{9}$$

where $t_{\max}$ is the maximal number of iterations. Those constants lead the exponential decay function radius to 1 when t reaches it maximum value, which is the maximal number of iterations [42].

With this proposed training, the network is able to adapt automatically to the presented input data. There is no need for an objective function as in supervised learning. The main disadvantage is related to the map size. For the training phase, the computational needs increase exponentially with the map size. The inference phase has lower computational needs compared to the training phase. The output is a map that depicts the used dataset as a representation in lower dimension.

3.3.4 Labeling phase

The goal of the labeling phase is to attribute additional information, such as the name, the color or the number of a cluster. Additional information relies on user's needs and is linked to the application. The generated map has observable clusters. Instead of identifying them manually, an automatic cluster identification phase has been created to do so.

The cluster identification phase wants to identify nodes that make up clusters and assign an information to the cluster to which they belong. The classification of map nodes is performed with Algorithm 2. It generates automatically clusters surrounded by boundaries, and an identifier is assigned to them. For example, if the node 43 with the coordinate (4,3) is localized inside the cluster "2," then this value is attributed to the node.

$$H_{ij} = \frac{w_{ij2} - \min_{(i,j)\in C} w_{ij2}}{\max_{(i,j)\in C} w_{ij2} - \min_{(i,j)\in C} w_{ij2}} \tag{10}$$

in which scale node values of each cluster are in the range $[0, 1]$. Those values represent the current state of the studied system. Each cluster has a degradation trend. For a node, a high HI value represents a healthy condition, whereas a low HI indicates a high degradation or a failure.

When a database with more than one fault mode is used to train the network, several subregions could appear on some clusters. Those subregions are linked to different fault modes related to part failures. To identify fault modes automatically, input data need to be labeled through the diagnostic phase.

The diagnostic phase aims at creating knowledge for the fault identification phase. The input data from the training set are again presented to the map. BMU searching provides the cluster number to which they belong and the associated HI. It leads to the input association using Algorithm 3. The building of the fault mode indication becomes possible by knowing exactly which sample appears in which clusters to localize occurring faults.

Algorithm 2. Cluster identification

```
For each Node on the Map not seen do
  | While Node is not a Boundary And has NeighborhoodNodes which are not Boundary do
  |   | Attribute ClusterLabel
  | End
  | Increment ClusterLabel
End
```

Algorithm 3. Diagnostic of input data

```
For each Sample in Input Data do
  | Present the sample to the map and localize the exited BMU node
  | Association to the sample : the cluster number and HI of the BMU node
End
```

A *Node* is considered as seen if it has been assigned to a cluster identifier, called *ClusterLabel*. *NeighborhoodNodes* are nodes that touch it in all four directions: up, down, left and right.

In the end, a map is generated in which cluster regions appear and are defined by boundaries surrounding them. Let us recall that weight vectors are associated with each node of the map. Then, for each cluster, a health index is built for nodes (i, j) that belong to cluster C by Eq. (10)

3.4 Fault modes identification

The identification of the fault modes gives an insight on the system state, such as the probability that a specific fault occurs, or its evolution through the time. Without prior knowledge about datasets, the number of faults is estimated through the fault quantification phase. Their area on the map is approximated thanks to a straightforward methodology based on probabilistic theory during the fault

subregions identification phase. Then, by presenting iteratively datasets with one associated fault to a map, that has one or more unknown fault, the fault modes association phase identifies the unknown faults.

3.4.1 Fault quantification

The fault quantification phase attempts to evaluate automatically the number of different system faults from a dataset (e.g., pressure drop and over-temperature are two different errors). Within a given time series of data referred to as cycles of a specific system, it can be assumed that the last cycle before non-recoverable error corresponds to the error state and can be used to identify faults [43]. At this cycle, the system is considered to be in a defect state and is taken out of service for maintenance actions. Before the stopping of the system, the advanced degradation of parts that were about to fail took place. This should be visible in the dataset. So, the last cycles of the system before breakdown are presented to the SOM and the labeling phase provides the best matching unit (BMU) (i.e., the hit node on the SOM map). The hit BMU can be the same for several instances of the system (for example, different aircraft jet engines belonging to the same family). The quantity of instances hitting this BMU indicates the hit number.

Two ways to estimate the hit number are now introduced:

- H1: using the last cycle of the system
- H2: using the last cluster hit of the system

In the first case, H1, there is only one last hit on a specific cluster for the system. For example, in a dataset with 249 systems, there are 249 last hits, shared by all map clusters, representing the final (most likely) faulty condition in which the system is found to be itself before it was stopped for maintenance. It represents a "sure" faulty condition.

In the second case, H2, the last hit for each system in each cluster is taken into consideration. Then, in the dataset with 249 systems, there are 249 last hits on each cluster. In the case of a six clusters map, this leads to 1494 last hits, representing faulty conditions for those operational conditions.

In the next section, it is shown that H2 provides more information and reliability than H1, and it is a decent approximation. H2 is used for the case study.

The frequency of those hits over each map cluster is linked to the fault number. Indeed, they tend to gather in areas that can be distinguished separately. Those hits are managed with tools from probability theory to build a representation of those subregions. A good candidate is the probability density function (PDF). The kernel density estimator [44] provides the estimation of the PDF such as:

$$\hat{f}_h(\vec{x}) = \frac{1}{n \cdot h} \sum_{i=1}^{N} K\left(\frac{\vec{x} - \overrightarrow{x_i}}{h}\right) \tag{11}$$

where $\vec{x} = (x^1, x^2, \ldots, x^p)$ are real values, $\overrightarrow{x_i}$ are random samples from an unknown distribution, N is the number of observation, K is the kernel smoothing function, which is a Gaussian kernel, and h is the bandwidth. In this study, the bandwidth has been selected at 1% of the SOM map size, leaving out of consideration the boundary nodes between fault clusters. For a map of 25 $\times$ 25 nodes without cluster, $h = 0.25$. According to the targeted application, the rule could evolve. Other kernel parameters are automatically estimated by the algorithm [45]. The PDF represents the probability distribution using the data samples where the kernel distribution sums the smoothing functions for each data value to produce a smooth, continuous probability curve. A 3D-PDF generation is used for each subregion, with node coordinate (x, y) and the hit number as a frequency as z coordinate. The generated function can be estimated at any (x, y) point. The number of peaks of the PDF leads to the number of faults inside each map cluster. Therefore, if a dataset has one or two fault modes, the method should lead, respectively, to one and two peaks for each cluster on the map. The goal of this approach is to be able to get an overview on the number of fault modes that are present in a dataset, without relying on a priori information.

3.4.2 Fault subregion identification

A cluster is surrounded by boundaries, and several small subregions can be estimated inside, related to fault modes. Fault subregions are the extracted regions from a cluster. They are generated from the separation of PDF peaks for a cluster and are used to define each fault area. Indeed, PDF uses Gaussian functions, which can be separated geometrically. However, all cluster nodes are not necessarily classified in a fault area. This is the case for nodes with weak PDF value, far away from the peak center. To address this problem, the PDF of a cluster is estimated at every cluster node. A custom threshold is applied on each estimated PDF, and fault areas are then generated with their own self-defined boundaries. The remaining nodes inside each fault area, after applying the threshold, represent the failure. So, if the node (4,3) is inside the subregion Failure 1, then this node is attributed to it.

3.4.3 Fault modes association

The association of fault modes (i.e., subregions of cluster) with a physical part is performed with similar data, which

present a known defect mode. Prior knowledge about fault modes, which is previously identified, is used. For example, a dataset with one fault mode is presented to a generated map that has been trained with a dataset with two fault modes. The presented dataset will excite nodes from one of the two subregions previously determined. This subregion will correspond to the known fault mode of the presented dataset.

4 Case study on aircraft jet engines

4.1 Overview

To illustrate the present work, a case study on diagnosing jet engines is used. Engine condition monitoring (ECM) allows for regular assessment of the jet engine health state, based on in-flight measured variables on the engine itself, as well as its environment (the aircraft) in its flight conditions. Specific parameter trend evolutions have shown to be early indications for engine degradations, failures and/or malfunctions [9]. Engine condition monitoring consists of a wide range of activities assessing the jet engine health, from the mounting on-wing until its removal. After every flight, performance engineers evaluate the evolution of engine critical parameters and derive from those analyses to anticipate or to avoid incidents, to evaluate the effects of incidents or to provide a clear "no problem for the next few flights" indication. Whenever an engine gets into a much deteriorated health state, no longer allowing operation within regulatory limits, the performance engineer recommends its removal and a precise planning. Actions of the performance engineer aim not only to keep the engine in its optimum operational condition, but also to correct in an early stage any detected malfunction, allowing for staying within safe operation and also reducing fuel consumption and increasing operational punctuality. Therefore, early fault mode identification provides relevant information for the maintenance program. The use of the SOM neural network for this application is particularly interesting for its ability to map the input data without prior knowledge on fault modes. In general, the monitored system does not provide labeled data related to fault modes, whereas in the case of only one fault mode, the situation is straightforward. In the case of multiple fault modes, it becomes more complicated without a proper monitoring to identify them.

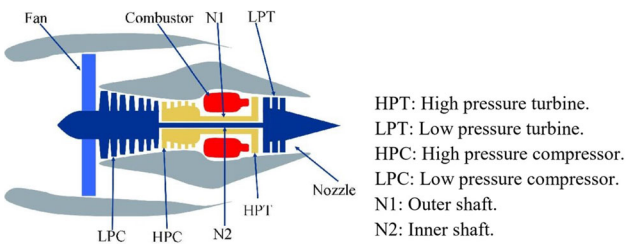


Fig. 4 Simplified diagram of the 90 K engine [46]

4.2 Input data

In this paper, datasets are generated [43] by using the C-MAPSS software [46]. C-MAPSS is a tool for the simulation of a realistic large commercial turbofan engine (Fig. 4) for the 90,000 lb thrust class. Thanks to editable input parameters, it is possible to specify operational profile, closed-loop controllers, and environmental conditions such as altitude. Furthermore, various degradations can be managed in different sections of the engine system.

Using this simulation environment, five datasets were generated by [43]. One of them was used for the prognostics challenge competition at International Conference on Prognostics and Health Management in 2008 (PHM08). In those datasets, the simulated engines have one or six operational conditions (flight phases such as Take-off, Cruise, etc.) driven by engine control settings (altitude, Mach number and Throttle Resolver Angle) and one or two fault modes. In PHM08 (Table 1), there are three datasets with one fault (i.e., #1, #2, and #5) and two with two faults (i.e., #3 and #4). The fault, corresponding to a failed system part, is, respectively, the HPC and the HPC and the fan (see Fig. 4). All dataset characteristics are summarized in Table 1.

Each dataset (i.e., #1 to #5) consists of multivariate time series and is divided into a training set and a testing set, generated by [43]. The database provides those sets in separate files: five training files and five testing files. The training subset is only used for training of the neural network (the learning), whereas the testing subset, with

Table 1 C-MAPSS dataset characteristics

Id	Name	Operational conditions	Fault modes	Failed system part	Number of engines
#1	FD001	1	1	HPC	100
#2	FD002	6	1	HPC	260
#3	FD003	1	2	HPC, Fan	100
#4	FD004	6	2	HPC, Fan	549
#5	FD005	6	1	HPC	218

Table 2 Output variables from C-MAPSS tool

Sensor id	Symbol	Description	Units
1	T2	Total temperature at fan inlet	°R
2	T24	Total temperature at LPC outlet	°R
3	T30	Total temperature at HPC outlet	°R
4	T50	Total temperature at LPT outlet	°R
5	P2	Pressure at fan inlet	psia
6	P15	Total pressure in bypass duct	psia
7	P30	Total pressure at HPC outlet	psia
8	Nf	Physical fan speed	rpm
9	Nc	Physical core speed	rpm
10	epr	Engine pressure ration (P50/P2)	–
11	Ps30	Static pressure at HPC outlet	psia
12	Phi	Ratio of fuel flow to Ps30	pps/psi
13	NRf	Corrected fan speed	rpm
14	NRc	Corrected core speed	rpm
15	BPR	Bypass ration	–
16	farB	Burner fuel–air ratio	–
17	htBleed	Bleed enthalpy	–
18	Nf_dmd	Demanded fan speed	rpm
19	PCNfR_dmd	Demanded corrected fan speed	rpm
20	W31	HPT coolant bleed	lbm/s
21	W32	LPT coolant bleed	lbm/s

Table 3 Extract of the dataset #4

Engine	Cycle	CS1	CS2	CS3	S1	S2	S3	S4	S5	S6	S7	S8	S9
1	1	42.0049	0.84	100	445	549.68	1343.43	1112.93	3.91	5.7	137.36	2211.86	8311.32
1	2	20.002	0.7002	100	491.19	606.07	1477.61	1237.5	9.35	13.61	332.1	2323.66	8713.6
1	3	42.0038	0.8409	100	445	548.95	1343.12	1117.05	3.91	5.69	138.18	2211.92	8306.69
1	4	42	0.84	100	445	548.7	1341.24	1118.03	3.91	5.7	137.98	2211.88	8312.35
1	5	25.0063	0.6207	60	462.54	536.1	1255.23	1033.59	7.05	9	174.82	1915.22	7994.94
1	6	34.9996	0.84	100	449.44	554.77	1352.87	1117.01	5.48	7.97	193.82	2222.77	8340
1	7	0.0019	0.0001	100	518.67	641.83	1583.47	1393.89	14.62	21.58	552.45	2387.92	9050.5
1	8	41.9981	0.84	100	445	549.05	1344.16	1110.77	3.91	5.69	137.13	2211.92	8307.28
1	9	42.0016	0.84	100	445	549.55	1342.85	1101.67	3.91	5.7	138.02	2211.9	8307.81
1	10	25.0019	0.6217	60	462.54	536.35	1251.91	1041.37	7.05	9.01	174.7	1915.23	8005.83
1	11	20.0016	0.7	100	491.19	606.88	1478.02	1233.07	9.35	13.61	333.22	2323.7	8709.62
1	12	34.9993	0.84	100	449.44	554.53	1365.99	1122.73	5.48	7.98	193.67	2222.78	8337.46
1	13	24.9986	0.62	60	462.54	536.32	1257.84	1040.87	7.05	9.01	174.53	1915.28	8000.07
1	14	20.0056	0.7008	100	491.19	607.32	1470.33	1242.41	9.35	13.61	333.71	2323.72	8714.35

Engine	Cycle	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21
1	1	1.01	41.69	129.78	2387.99	8074.83	9.3335	0.02	330	2212	100	10.62	6.367
1	2	1.07	43.94	312.59	2387.73	8046.13	9.1913	0.02	361	2324	100	24.37	14.6552
1	3	1.01	41.66	129.62	2387.97	8066.62	9.4007	0.02	329	2212	100	10.48	6.4213
1	4	1.02	41.68	129.8	2388.02	8076.05	9.3369	0.02	328	2212	100	10.54	6.4176

Table 3 (continued)

Engine	Cycle	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21
1	5	0.93	36.48	164.11	2028.08	7865.8	10.8366	0.02	305	1915	84.93	14.03	8.6754
1	6	1.02	41.44	181.9	2387.87	8054.1	9.3346	0.02	330	2223	100	14.91	8.9057
1	7	1.3	46.94	520.48	2387.89	8127.92	8.396	0.03	391	2388	100	38.93	23.4578
1	8	1.01	41.6	129.65	2387.97	8075.99	9.3679	0.02	329	2212	100	10.55	6.2787
1	9	1.02	41.44	129.65	2388	8071.13	9.3384	0.02	328	2212	100	10.63	6.3055
1	10	0.94	36.24	164.08	2028.13	7869.41	10.9141	0.02	305	1915	84.93	14.34	8.6119
1	11	1.07	43.86	312.96	2387.83	8050.06	9.1667	0.02	363	2324	100	24.63	14.6705
1	12	1.02	41.45	181.71	2387.86	8056.31	9.3041	0.02	332	2223	100	14.68	8.8752
1	13	0.94	36.42	163.67	2028.14	7865.15	10.8388	0.02	305	1915	84.93	14.41	8.6062
1	14	1.07	43.92	313.3	2387.85	8051.34	9.2272	0.02	364	2324	100	24.3	14.7105

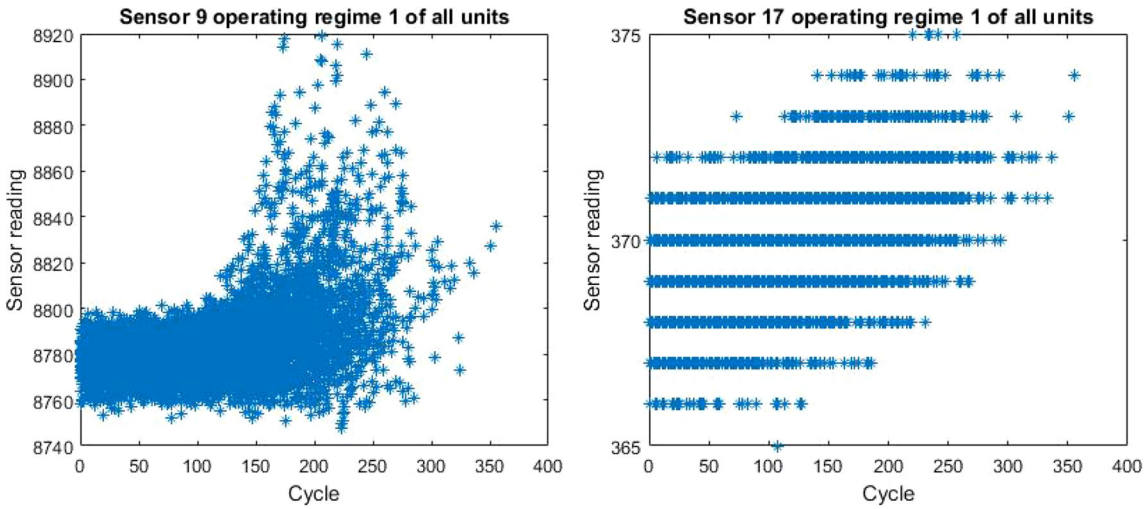


Fig. 5 Trend of sensors in a selected regimes. (Left) Inconsistent end-life trends. (Right) Piecewise trends

different data, is only used for the network validation. In this case study, a dataset contains three input variables representing the engine operational settings, that generate one or six operational conditions and 21 output sensors (Table 2). Dataset is comparable between themselves whether they have the same operation settings, generating the same number of operational conditions. Thus, FD001 and FD003 are comparable as well as FD002 and FD004. However, the FD005 dataset cannot be compared with FD002 or FD004 because the values of the operational settings are not compatible, even if it has six operational conditions.

Table 3 shows an extract of available data for the dataset #4. The other datasets follow the same format.

A reduction in the number of sensors could lead to a drastic reduction in the computational time for the training of the neural network. Following the work of [10, 22, 47] for the PHM08 dataset number #5, the only relevant seven sensors were found to be: 2, 3, 4, 7, 11, 12 and 15.

In fact, among the 21 sensors, constant or binary trend is observed on several sensors that do not provide degradation behaviors. Sensors 1, 5, 6, 10, 16, 17, 18 and 19 are concerned and not considered from the selection. Others sensors provide similar information such as sensors 8 and 13

Table 4 SOM information for all datasets

Id	Number of observations	Features	Map size	Iterations[a]
#1	20,631	3	7	24,500
#2	53,759	3	9	40,500
#3	24,720	3	7	24,500
#4	61,249	3	9	40,500
#5	45,918	3	8	32,000

[a]Estimated following the rule of thumb

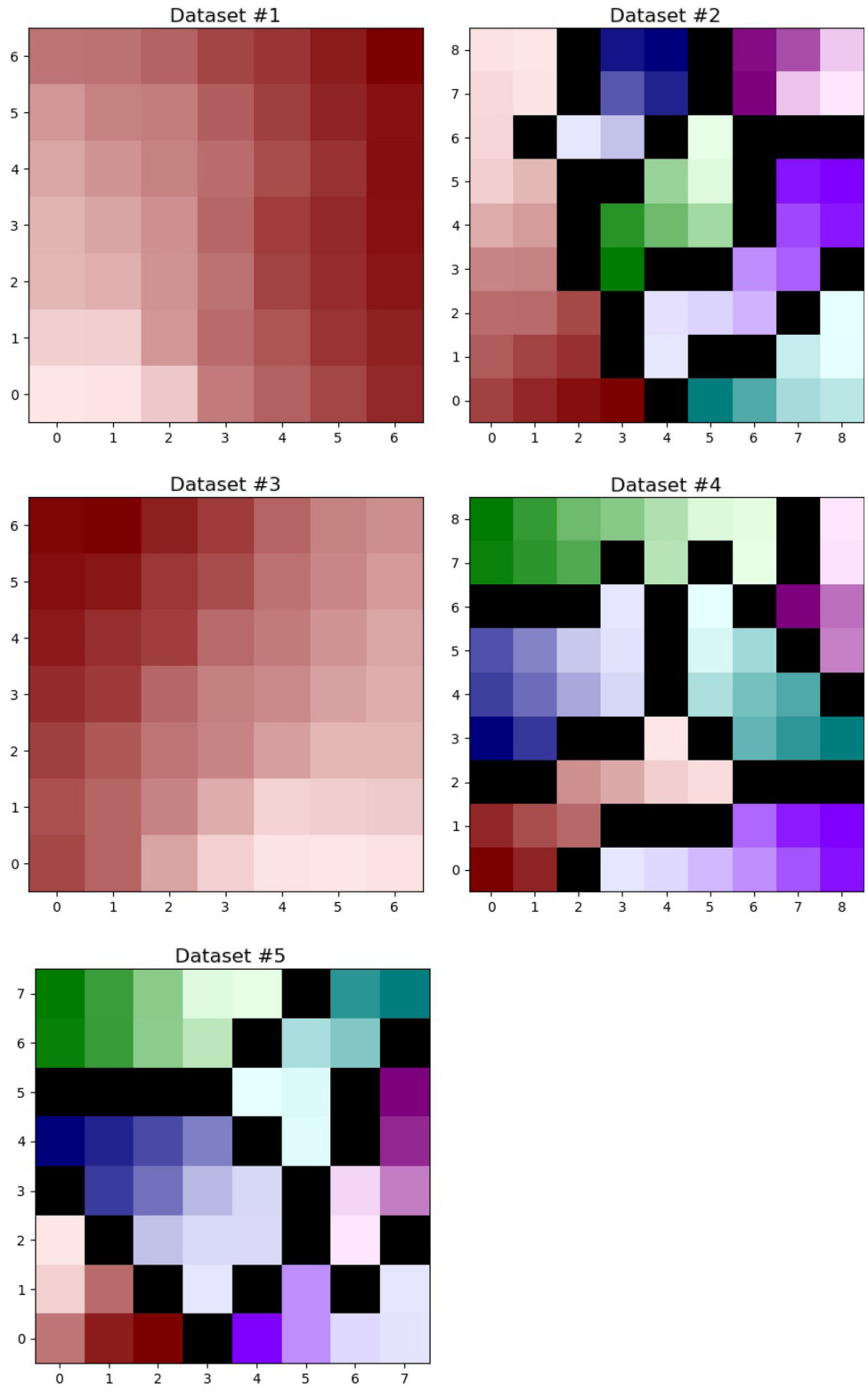


Fig. 6 SOM maps from datasets #1 to #5 with three operation conditions as features

with the sensor 11 by looking at the correlation coefficient with a threshold of 85%. Sensors 9 and 14 show inconsistent end-life trends among the engines (Fig. 5, left), and the sensor 17 is a piecewise constant function (Fig. 5, right). Finally, the sensors 20 and 21 do not bring a clear trend throughout the unit's life according to [22]. Through those steps, the final seven sensors are determined. The

Table 5 Training SOM information for all datasets

Id	Number of observations	Features	Map size	Iterations[a]
#1	20,631	7	19	180,500
#2	53,759	7	24	288,000
#3	24,720	7	20	200,000
#4	61,249	7	25	312,500
#5	45,918	7	23	264,500

[a]Estimated following the rule of thumb

same selected sensors have been taken into account for the datasets #1, #2, #3 and #4.

4.3 Operational mode labeling

The case study contains five datasets generated by [43] (see Table 1), where it is known that one or six different operational conditions are used. According to the complexity of the case study, manual operational mode labeling may not be possible by hand. To demonstrate the power of the automated SOM, they are performed following the system map process introduced in the previous section (see Fig. 2).

The three operational settings (altitude, Mach number and Throttle Resolver Angle) in the dataset are used as inputs to the SOM. Due to the number of inputs, the map size is determined with Eq. (3) and the result is divided by four, custom factors established through multiple empirical experiments. A bigger (or smaller) map leads to an increase (and decrease) in the cluster numbers and may lead to an inconsistency in the representation of information. Further development will be done to address this empirical estimation. The convergence criteria of the "rule of thumb" are used, leading to a number of iterations of 500 times the number of neurons. Table 4 summarizes SOM information used.

The training phase generates maps in Fig. 6.

The maps reveal one cluster for the datasets #1 and #3, whereas the datasets #2, #4 and #5 show six clusters. This means that there is one operational condition for #1 and #3 and six operational conditions for #2, #4 and #5, which is in line with Table 4. The diagnostic phase from the system map process provides exactly the same operational mode labeling as what was obtained manually.

4.4 System map generation

4.4.1 Preprocessing and training

The SOM will now be trained with the seven sensors identified in the previous section. The number of neurons is determined with Eq. (3) and divided by two to reduce the computational time. Table 5 summarizes SOM information used.

The training phase generates the maps in Fig. 7.

For each dataset in Fig. 7, the number of clusters is identical to Fig. 6, corresponding to the number of operational modes. The cluster identification phase provides the same cluster information with the seven selected sensors, compared to the operational mode labeling in Sect. 4.3. It provides reliability in the unsupervised approach.

It appears that the maps for the datasets #3 and #4 show two darker colors on each cluster, which means that there are two fault modes in each cluster. This matches the information of Table 1. The color degradation corresponds to the evolution of the HI (i.e., degradation status). Lighter colors correspond to a healthy system, whereas the darker colors mean an advanced degradation. Other datasets have only one dark colors on each cluster; they have one fault mode. To confirm that, mathematical tools are now introduced.

4.5 Fault mode identification

4.5.1 Fault quantification

The kernel density estimator Eq. (11) is applied on each cluster of each map under H2. It generates a PDF to identify the number of fault modes. Thanks to the SOM map (Fig. 7), the minimum probability density function kernel bandwidth without cluster boundaries can be obtained (see Table 6).

Figure 8 presents the PDF generated for a particular cluster. On those figures, PDF values of node coordinates (x, y) and hit numbers z are normalized. The number of fault modes is easily identifiable visually as well as automatically. This procedure is performed for each generated cluster on each map, and results are compared with the information provided by the datasets. It results that the fault number is well identified, with two fault modes for the database #3, #4 and one for the others.

In Sect. 3.4.1, two ways to evaluate the hit number were presented. Here, we would like to evaluate whether H1 is more pertinent than H2. H1 is relevant in terms of interpretation. However, the dataset used provides few engines (Table 1). Table 7 summarizes the number of samples according to hypotheses presented in Sect. 3.4.1.

Under hypothesis H1, due to a lack of samples, only one out of two fault modes is identified. For dataset #4, it has two fault modes and six operational conditions; the 249 samples of H1 represent around 21 samples per fault per operational conditions. Following the philosophy of H2, around 100 samples per fault per operational conditions are needed for proper identification.

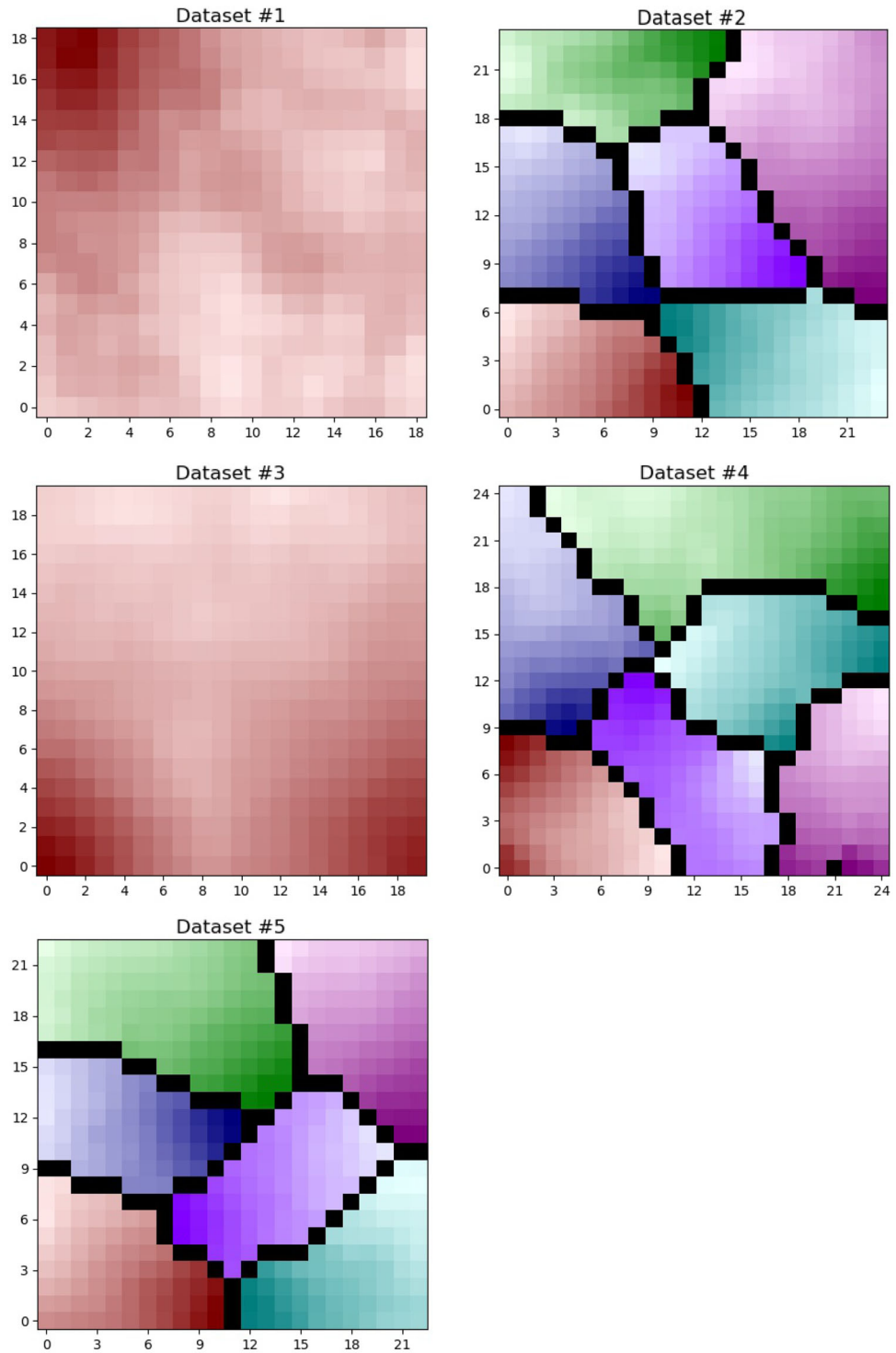


Fig. 7 SOM map from datasets #1 to #5 with 7 physical sensor data as features

For each engine, the last flight cycle spent on each operational condition (H2) is compared to the last flight cycle before a fault occurs (H1) to quantify the reliability of H2. For example, the engine 12 (Table 8) has six operational conditions. Table 8 summarizes the flight cycle for both cases and the error of H2 compared to H1. The

Table 6 PDF kernel bandwidth

Id	Map size	Bandwidth h
#1	19	0.19
#2	24	0.22
#3	20	0.20
#4	25	0.22
#5	23	0.21

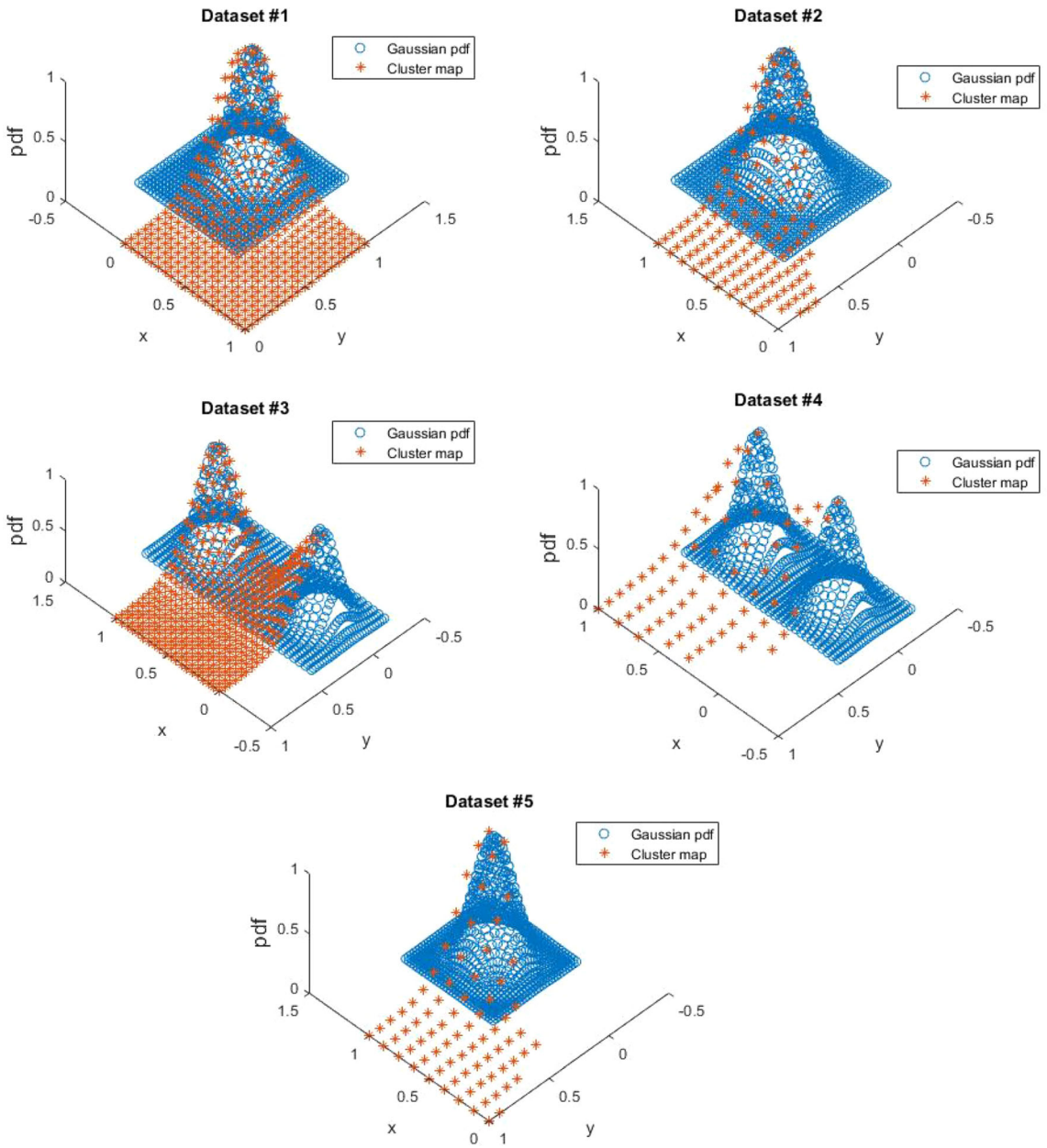


Fig. 8 PDF on a cluster for datasets #1 to #5

Table 7 Number of sample for datasets #1 to #5

Id	H1	H2
#1	100	100
#2	260	1560
#3	100	100
#4	249	1494
#5	218	1308

error is above 10% (custom threshold) for the clusters 1 and 5. That means engine 12 belongs to the group of engines, where H1 is more relevant than H2.

Table 8 Example of comparison for engine 12

Engine 12			
Cluster	H1	H2	Error (%)
1	320	260	18.75[a]
2		320	0.00
3		300	6.25
4		289	9.69
5		277	13.44[a]
6		310	3.13

[a]Error above 10%

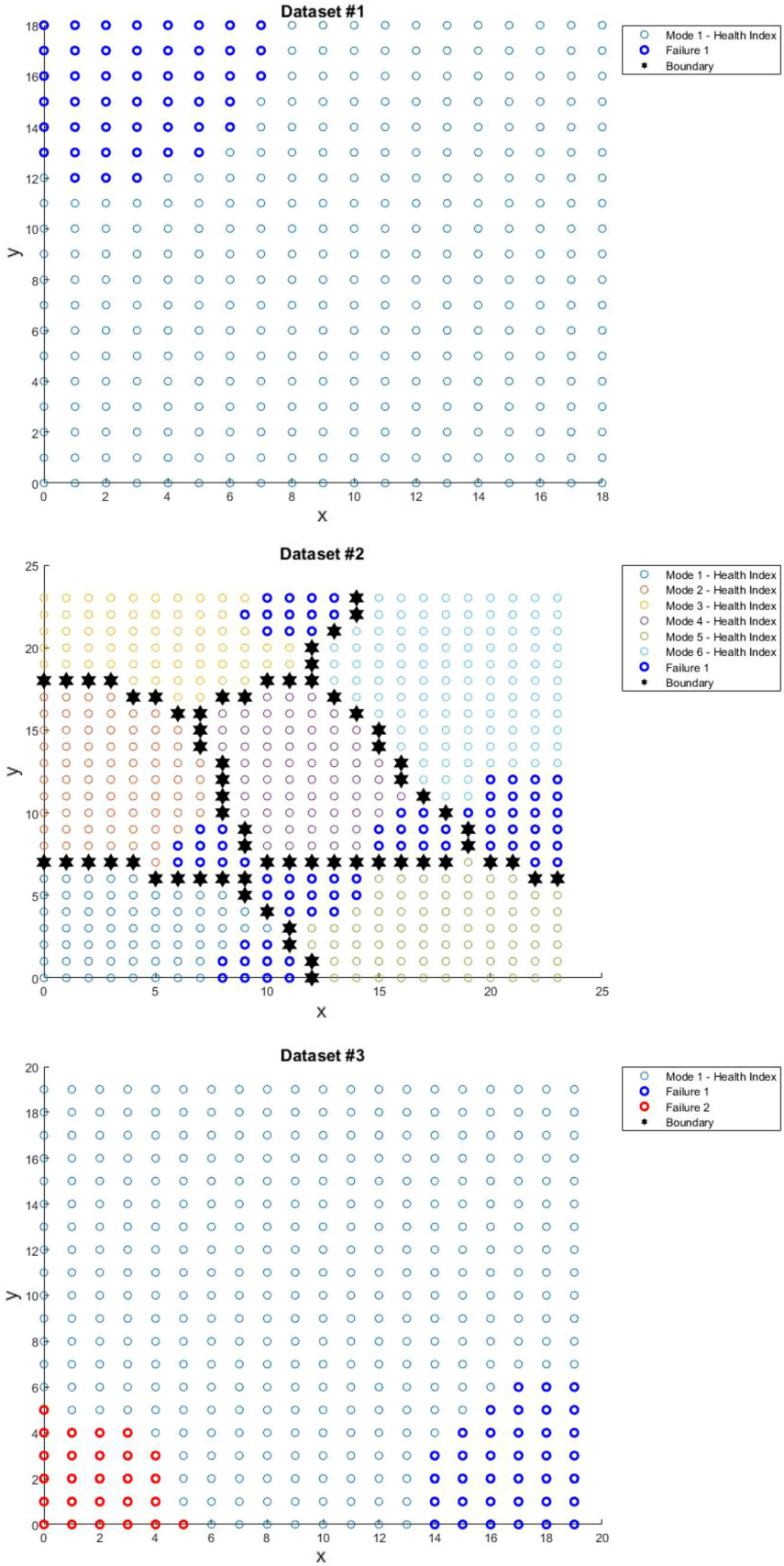
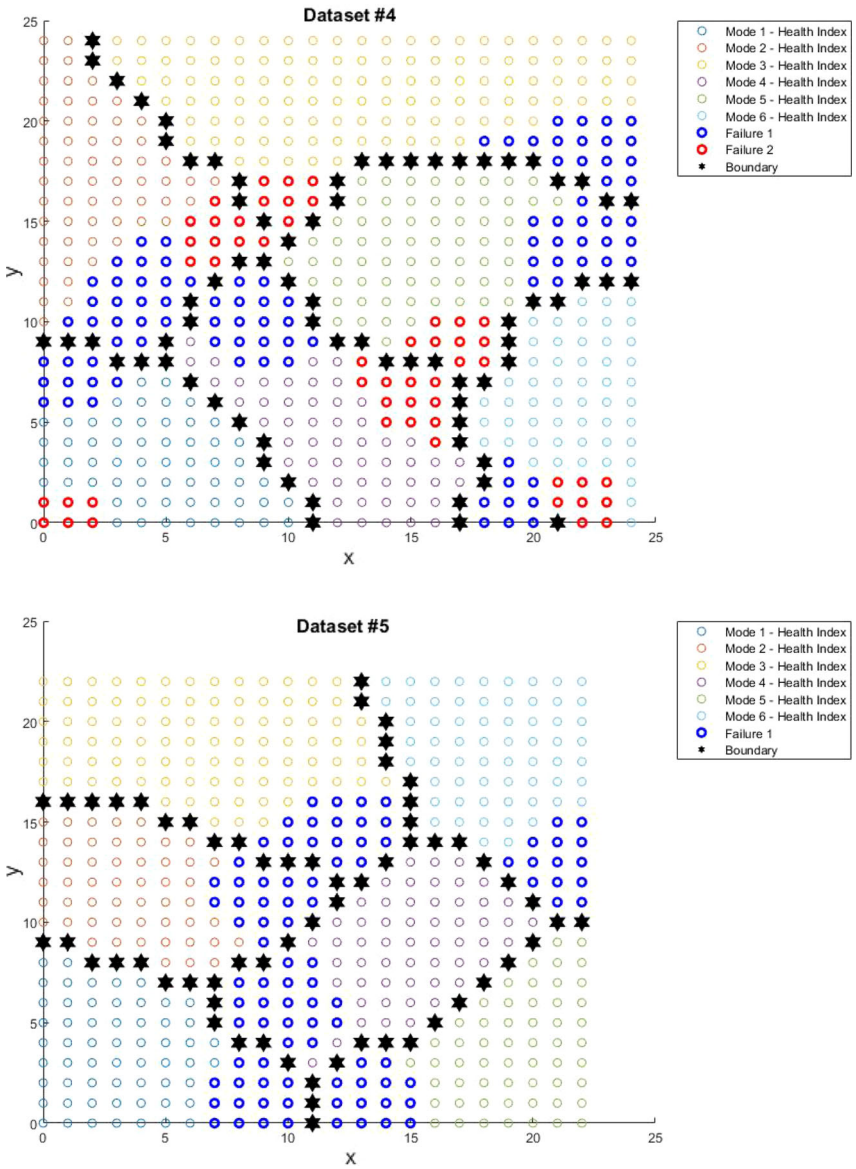


Fig. 9 Failure modes for datasets #1 to #5

Fig. 9 continued



This evaluation is performed for all datasets with a custom threshold of 10%. It results that there are around 3% of engines where H2 is not relevant. For a dataset of 249 engines, H1 is more relevant than H2 for only eight engines. H2 is therefore acceptable for this case study.

4.5.2 Fault subregions identification

For the fault subregions identification, only map nodes that are included in the PDF shape are retained. The user customizes the threshold according to the required precision. This is linked to the probability of training engines of which their last flight cycle has hit the fault mode area. A threshold of 0.40 was used, meaning that all nodes with a probability density lower than 40% are removed. The results are shown in Fig. 9.

When an engine degrades up to a point a failure mode is likely to happen, fault mode area is crossed on each operational mode. Based on this observation, the fault mode is considered to be the same on each map cluster and labeled as failure in Fig. 9.

4.5.3 Fault modes association

The fault modes association phase can only be performed on similar datasets as explained in Sect. 4.2. Thus, the datasets #1 and #3 are similar (similar operational settings and operation conditions) as well as the datasets #2 and #4. The use of datasets with one fault mode (i.e., the datasets #1 and #2) to identify the same failure on the maps trained with two failures (i.e., the datasets #3 and #4) results in a

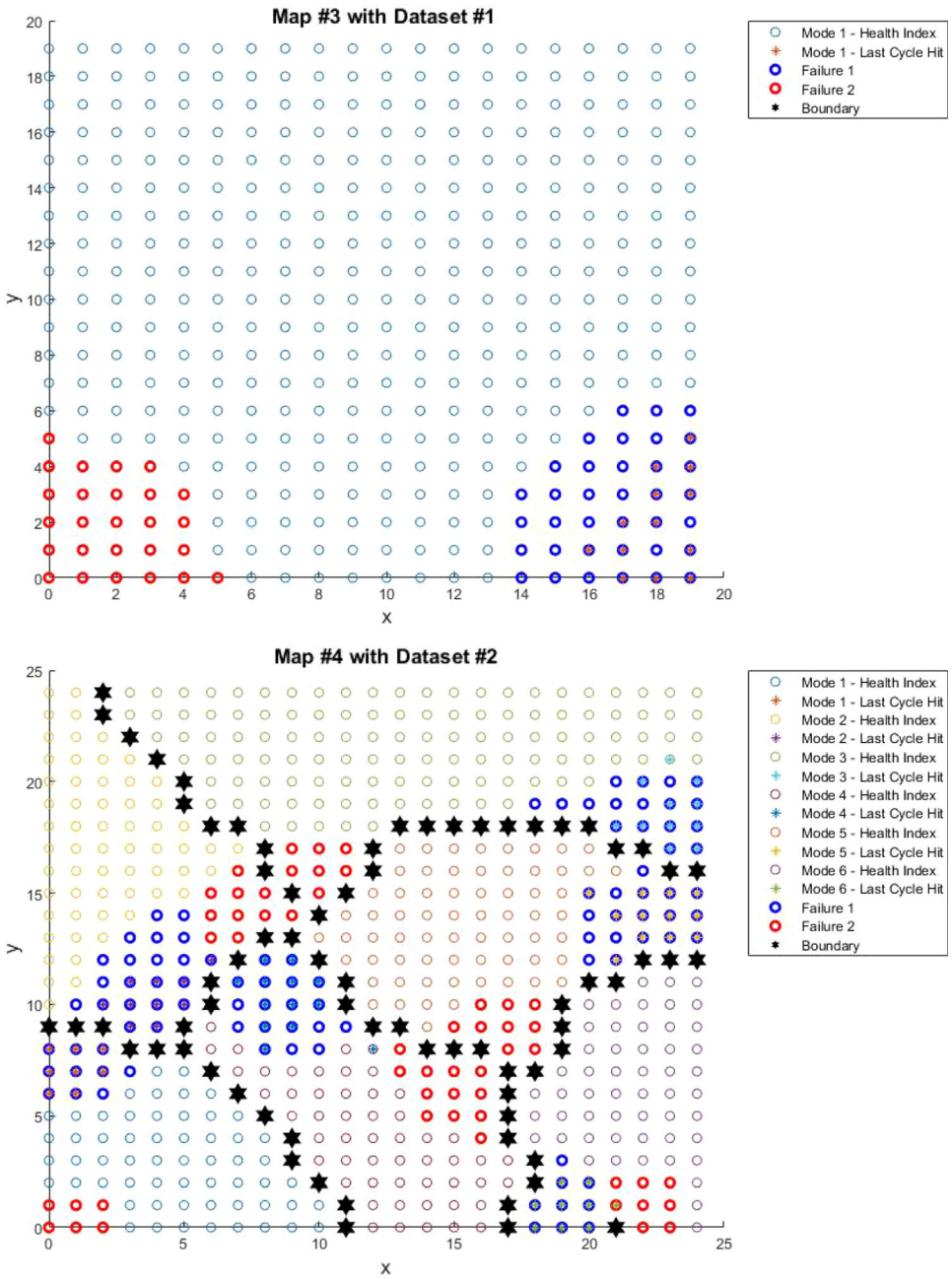


Fig. 10 Fault modes identification for maps #3 and #4

clear and unambiguous identification. The found fault mode is associated with the corresponding failure.

As shown in Fig. 9, the datasets #3 and #4 have both two fault modes, named, respectively, failures 1 and 2. Yet, the faulty system part is unknown. As mentioned, datasets #1 and #2 are used to identify one of the two failures on maps trend with datasets #3 and #4.

Figure 10 shows that all engines from the datasets #1 and #2 are, respectively, in the cluster failure 1 of the datasets #3 and #4, following the hit number estimation 'H2' (see Sect. 3.4.1). Knowing that #1 and #2 have HPC fault modes (Table 1), failures 1 and 2 are, respectively, identified as HPC and fan fault mode. With this knowledge, the fault modes of each engine in the datasets #3 and #4 can be determined.

However, on the map #4 (Fig. 10), some engines are out of a failure area, such as in modes 3 and 4, or misclassified, such as in mode 6. With the hit number estimation 'H2,' it represents 0.19% of error, whereas with the hit number estimation 'H1,' all engines are perfectly classified for this study. The failure identification procedure is then considered satisfactory.

5 Conclusion

This study addresses early detection and classification of faults through an unsupervised learning approach, without prior knowledge, based on self-organizing maps (SOMs). This neural network is at the core of the automatized approach. A complete process to comprehend the concept and to use SOM has been presented. The SOM has some advantage such as unsupervised training and is useful for dimensionality reduction and representation of complex and large datasets thanks to the map visualization. A methodology has been described to build and to configure the SOM according to the case study. The article highlights the possibility to identify the operational mode and fault modes inside generated maps with a methodology relying on the kernel density estimation. The methodology has been illustrated on a case study for diagnosing jet engine datasets. Without prior knowledge on the faults, the proposed algorithm was able to identify the number of operational modes as well as the fault mode number for the five datasets. Furthermore, the fault subregions identification estimates fault mode areas on the map, leading to failures identification on each cluster.

The unsupervised classifier used is a SOM neural network. There exist different types of unsupervised clustering techniques such as hierarchical or Bayesian clustering that could provide different results according to the case study [48]. Another candidate for future work could be a network based on restricted Boltzmann machines (RBM) [49] that will provide a probability distribution over its set of inputs.

In the current study, the automatic fault mode identification was experimented up to two fault modes. For future work, the study should be extended to more than two failure modes on the same case study. Two ways to evaluate the hit number have been explored. Other hypothesis could be examined. The used feature for #1 to #4 was supposed to be same than for #5. A generic approach to get automatically the best set of feature for different types of data will be a good solution to consider, as well as for the custom factor for the map size reduction. A study could be performed to explore the use of the presented approach with a different case study.

Acknowledgements The author affiliated to Sogeti High Tech and ISAE-SUPAERO gratefully acknowledges his colleagues who provided insight and expertise through this paper.

Compliance with ethical standards

Conflict of interest Sébastien Schwartz, Juan José Montero Jimenez, Michel Salaün and Rob Vingerhoeds declare that they have no conflict of interest.

References

1. Simon DL, Rinehart AW (2014) A model-based anomaly detection approach for analyzing streaming aircraft engine measurement data. In: Volume 6: Ceramics; controls, diagnostics and instrumentation; education; manufacturing materials and metallurgy, Düsseldorf, Germany. ASME, p V006T06A032. https://doi.org/10.1115/gt2014-27172
2. Naderi E, Meskin N, Khorasani K (2012) Nonlinear fault diagnosis of jet engines by using a multiple model-based approach. J Eng Gas Turbines Power 134(1):011602. https://doi.org/10.1115/1.4004152
3. Zeng D, Zhou D, Tan C, Jiang B (2018) Research on model-based fault diagnosis for a gas turbine based on transient performance. Appl Sci 8(1):148. https://doi.org/10.3390/app8010148
4. Naderi E, Khorasani K (2017) Data-driven fault detection, isolation and estimation of aircraft gas turbine engine actuator and sensors. In: 2017 IEEE 30th Canadian conference on electrical and computer engineering (CCECE), Windsor, ON. IEEE, pp 1–6. https://doi.org/10.1109/ccece.2017.7946715
5. Sarkar S, Jin X, Ray A (2011) Data-driven fault detection in aircraft engines with noisy sensor measurements. J Eng Gas Turbines Power 133(8):081602. https://doi.org/10.1115/1.4002877
6. Svärd C, Nyberg M, Frisk E, Krysander M (2014) Data-driven and adaptive statistical residual evaluation for fault detection with an automotive application. Mech Syst Signal Process 45(1):170–192. https://doi.org/10.1016/j.ymssp.2013.11.002
7. Pourbabaee B, Meskin N, Khorasani K (2016) Robust sensor fault detection and isolation of gas turbine engines subjected to time-varying parameter uncertainties. Mech Syst Signal Process 76–77:136–156. https://doi.org/10.1016/j.ymssp.2016.02.023
8. Venkatasubramanian V (2005) Prognostic and diagnostic monitoring of complex systems for product lifecycle management: challenges and opportunities. Comput Chem Eng 29(6):1253–1263. https://doi.org/10.1016/j.compchemeng.2005.02.026
9. Vingerhoeds RA, Janssens P, Netten BD, Aznar Fernández-Montesinos M (1995) Enhancing off-line and on-line condition monitoring and fault diagnosis. Control Eng Pract 3(11):1515–1528. https://doi.org/10.1016/0967-0661(95)00162-N
10. Montero Jimenez JJ, Vingerhoeds R (2018) Enhancing operational fault diagnosis by assessing multiple operational modes. In: MOSIM'18—Conférence Internationale de Modélisation, Optimisation et Simulation, Toulouse
11. Kohonen T (1982) Self-organized formation of topologically correct feature maps. Biol Cybern 43(1):59–69. https://doi.org/10.1007/BF00337288
12. Germen E, Başaran M, Fidan M (2014) Sound based induction motor fault diagnosis using Kohonen self-organizing map. Mech Syst Signal Process 46(1):45–58. https://doi.org/10.1016/j.ymssp.2013.12.002
13. Côme E, Cottrell M, Verleysen M, Lacaille J (2010) Aircraft engine health monitoring using self-organizing maps. In: Perner P (ed) Advances in data mining. Springer, Berlin, pp 405–417. https://doi.org/10.1007/978-3-642-14400-4_31
14. Cottrell M, Gaubert P, Eloy C, François D, Hallaux G, Lacaille J, Verleysen M (2009) Fault prediction in aircraft engines using self-organizing maps. In: Príncipe JC, Miikkulainen R (eds) Advances in self-organizing maps, vol 5629. Springer, Berlin, pp 37–44. https://doi.org/10.1007/978-3-642-02397-2_5
15. Katunin A, Amarowicz M, Chrzanowski P (2015) Faults diagnosis using self-organizing maps: a case study on the

DAMADICS benchmark problem, pp 1673–1681. https://doi.org/10.15439/2015f26
16. Yu H, Khan F, Garaniya V (2015) Risk-based fault detection using self-organizing map. Reliab Eng Syst Saf 139:82–96. https://doi.org/10.1016/j.ress.2015.02.011
17. Chen X, Yan X (2012) Using improved self-organizing map for fault diagnosis in chemical industry process. Chem Eng Res Des 90(12):2262–2277. https://doi.org/10.1016/j.cherd.2012.06.004
18. Dharshini R, Hemanandhini S (2016) Brain tumor segmentation based on self organising map and discrete wavelet transform. In: 2016 international conference on computer communication and informatics (ICCCI), Coimbatore, India. IEEE, pp 1–9. https://doi.org/10.1109/iccci.2016.7479960
19. Peel L (2008) Data driven prognostics using a Kalman filter ensemble of neural network models. In: 2008 international conference on prognostics and health management. https://doi.org/10.1109/phm.2008.4711423
20. Jolliffe I (2011) Principal component analysis. Springer, Berlin. https://doi.org/10.1007/b98835
21. Sammon JW (1969) A nonlinear mapping for data structure analysis. IEEE Trans Comput 100(5):401–409. https://doi.org/10.1109/t-c.1969.222678
22. Wang T, Jianbo Y, Siegel D, Lee JA (2008) Similarity-based prognostics approach for remaining useful life estimation of engineered systems. In: 2008 international conference on prognostics and health management. IEEE, pp 1–6. https://doi.org/10.1109/phm.2008.4711421
23. Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3:1157–1182
24. Jovic A, Brkic K, Bogunovic N (2015) A review of feature selection methods with applications. In: 2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO), Opatija, Croatia. IEEE, pp 1200–1205. https://doi.org/10.1109/mipro.2015.7160458
25. Saeys Y, Inza I, Larranaga P (2007) A review of feature selection techniques in bioinformatics. Bioinformatics 23(19):2507–2517. https://doi.org/10.1093/bioinformatics/btm344
26. Visalakshi S, Radha V (2014) A literature review of feature selection techniques and applications: review of feature selection in data mining. In 2014 IEEE international conference on computational intelligence and computing research. IEEE, Coimbatore, India, pp 1–6. https://doi.org/10.1109/iccic.2014.7238499
27. Kohonen T (1997) Springer series in information sciences, vol 30. Springer, Berlin. https://doi.org/10.1007/978-3-642-97966-8
28. Kohonen T (2014) Unigrafia, Helsinki, Finland
29. Zin ZM (2014) Cluster and visualize data using 3D self-organizing maps. In: 2014 11th international conference on ubiquitous robots and ambient intelligence (URAI). IEEE, pp 163–168. https://doi.org/10.1109/urai.2014.7057523
30. Azcarraga A, Manalili S (2011) Design of a structured 3D SOM as a music archive, pp 188–197. https://doi.org/10.1007/978-3-642-21566-7_19
31. Gorricha J, Lobo V (2012) Improvements on the visualization of clusters in geo-referenced data using self-organizing maps. Comput Geosci 43:177–186. https://doi.org/10.1016/j.cageo.2011.10.008
32. El Tobely T, Salem A (2005) Position detection of unexploded ordnance from airborne magnetic anomaly data using 3-D self organized feature map. In: Proceedings of the fifth IEEE international symposium on signal processing and information technology, 2005. IEEE, pp 322–327. https://doi.org/10.1109/isspit.2005.1577117
33. Fujimura K, Masuda K, Fukui Y (2006) A consideration on the multi-dimensional topology in self-organizing maps. In: 2006 international symposium on intelligent signal processing and communications, IEEE, pp 825–828. https://doi.org/10.1109/ispacs.2006.364772
34. Côme E, Cottrell M, Verleysen M, Lacaille J (2010) Self organizing star (SOS) for health monitoring. In: European conference on artificial neural networks, pp 99–104
35. Tian J, Azarian MH, Pecht M (2014) Anomaly detection using self-organizing maps-based k-nearest neighbor algorithm. In: European conference of the prognostics and health management society
36. Engelbrecht AP (2007) Computational intelligence: an introduction, 2nd edn. Wiley, Hoboken. https://doi.org/10.1002/9780470512517
37. Su M-C, Liu T-K, Chang H-T (1999) An efficient initialization scheme for the self-organizing feature map algorithm. In: IJCNN'99. International joint conference on neural networks. Proceedings (Cat. No. 99CH36339), vol 3. IEEE, pp 1906–1910. https://doi.org/10.1109/ijcnn.1999.832672
38. Kohonen T (2001) Springer series in information sciences, vol 30, 3rd edn. Springer, Berlin. https://doi.org/10.1007/978-3-642-56927-2
39. Kaski S, Venna J, Kohonen T (2000) Coloring that reveals cluster structures in multivariate data. Aust J Intell Inf Process Syst 6:82–88
40. Alahakoon D, Halgamuge SK, Srinivasan B (2000) Dynamic self-organizing maps with controlled growth for knowledge discovery. IEEE Trans Neural Netw 11(3):601–614. https://doi.org/10.1109/72.846732
41. Natita W, Wiboonsak W, Dusadee S (2016) Appropriate learning rate and neighborhood function of self-organizing map (SOM) for specific humidity pattern classification over Southern Thailand. Int J Model Optim 6(1):61–65. https://doi.org/10.7763/IJMO.2016.V6.504
42. Zhang W, Wang J, Jin D, Oreopoulos L, Zhang Z (2018) A deterministic self-organizing map approach and its application on satellite data based cloud type classification. In: Conference IEEE Big Data
43. Saxena A, Goebel K, Simon D, Eklund N (2008) Damage propagation modeling for aircraft engine run-to-failure simulation. In: 2008 international conference on prognostics and health management. IEEE, pp 1–9. https://doi.org/10.1109/phm.2008.4711414
44. Kafadar K, Bowman AW, Azzalini A (1999) Applied smoothing techniques for data analysis: the kernel approach with S-PLUS. J Am Stat Assoc 94(447):982. https://doi.org/10.2307/2670015
45. Bowman AW, Azzalini A (1997) Applied smoothing techniques for data analysis: the kernel approach with S-Plus illustrations, vol 18. Oxford University Press, Oxford
46. Frederick DK, DeCastro JA, Litt JS (2007) User's guide for the commercial modular aero-propulsion system simulation (C-MAPSS)
47. Hu C, Youn BD, Wang P, Taek Yoon J (2012) Ensemble of data-driven prongostic algorithms for robust prediction of remaining useful life. Reliab Eng Syst Saf 103:120–135. https://doi.org/10.1016/j.ress.2012.03.008
48. Fusco G, Perez J (2019) Bayesian network clustering and self-organizing maps under the test of Indian Districts. A comparison. Cybergeo. https://doi.org/10.4000/cybergeo.31909
49. Zhang X, Yao L, Wang X, Monaghan J, Mcalpine D, Zhang Y (2019) Cs Eess Q-Bio. arXiv:1905.04149