Machine Learning-Based Multivariate Time Series Analysis For Health Monitoring And Prognostics Of Complex Systems

Etienne Jules

UNIVERSITÉ CLERMONT AUVERGNE

École Doctorale Sciences Pour l'Ingénieur de Clermont-Ferrand, ED SPI

PHD THESIS MANUSCRIPT

Machine Learning-Based Multivariate Time Series Analysis for Health Monitoring and Prognostics of Complex Systems

—

Etienne JULES

Université Clermont Auvergne, Clermont Auvergne INP, CNRS, Institut Pascal

Presented and publicly defended on February 13, 2024 in the presence of the following jury members :

President :

Engelbert Mephu-Nguifo Professor, Université Clermont Auvergne

Rapporteurs :

Piero Baraldi Professor, Politecnico di Milano, Italy

Christian Gogu Professor, ISAE-Supaéro

Examiners :

Zeina Al Masry Associate professor, FEMTO-ST Institute

Bruno Sudret Professor, ETH Zurïch, Switzerland

Thesis director and co-supervisor:

Jean-Marc Bourinet Professor, Université Clermont Auvergne

Co-supervisor:

Cécile Mattrand Associate professor, Université Clermont Auvergne

Résumé en Français

Avec les progrès rapides des technologies de mesure, l'acquisition continue de données de monitoring est devenue possible. Ces données se présentent principalement sous la forme de séries temporelles multivariées (STM). L'accès à ces STM surveillées sur de longues périodes de la vie d'un système complexe offre de nombreuses possibilités pour développer des connaissances approfondies sur la dynamique sous-jacente de ce système. En ingénierie, le domaine du *prognostic and health monitoring* (PHM), peut bénéficier de l'accès à ces STM. Le PHM vise à garantir des conditions de fonctionnement satisfaisantes des systèmes physiques tout au long de leur durée de vie. Dans cette thèse, les tâches spécifiques du PHM liées à l'apprentissage des STM sont étudiées. Les méthodes d'apprentissage automatique appliquées aux SCM sont explorées pour résoudre ces tâches. L'accent est mis sur les réseaux de neurones en raison de leurs progrès récents et de leurs performances dans divers domaines d'application. Parmi les quatre tâches du PHM, à savoir : la détection des anomalies, le pronostic de la durée de vie restante (RUL), l'identification des modes de défaillance et l'évaluation de l'état de santé, cette dernière est étudiée plus en détail dans cette thèse. Elle implique la construction d'un indicateur d'état de santé (HI), c'est-à-dire une grandeur évoluant dans le temps et indiquant l'état de dégradation du système. Il est en effet avancé dans ce manuscrit que cette tâche est centrale pour le PHM et qu'elle peut aider à résoudre ses autres tâches plus spécifiques. Une nouvelle méthode d'apprentissage du HI utilisant l'apprentissage par similarité avec un réseau de neurones siamois et une fonction perte par contraste est proposée. Cette méthode est spécifiquement conçue pour favoriser la monotonie des HI, qui est une propriété cruciale que ces dernièrs doivent satisfaire. Cette méthode est testée sur deux jeux de données publiques fournissant des signaux provenant respectivement de turbines d'avion et de roulements à billes. Ces expériences valident la conception de notre méthode pour contraindre la monotonie du HI et montrent une amélioration par rapport aux travaux existants. La méthode proposée fournit donc des HI monotones, mais aussi un espace de représentation latent où les échantillons des signaux mesurés peuvent être projetés. Cet espace latent est de faible dimension et représente typiquement des échantillons proches les uns des autres s'ils partagent des niveaux de dégradation similaires. Sur la base de cette nouvelle approche d'apprentissage du HI, deux méthodes sont proposées pour prédire le RUL, c'est-à-dire le temps qu'il reste à une instance du système étudié pour entrer en défaillance. La première méthode est basée sur la prédiction des futures valeurs de HI couplée à une estimation du seuil de défaillance, toutes deux réalisées dans le cadre d'une approche intégrée combinant des méthodes issues des processus stochastiques et des réseaux de neurones. La deuxième méthode proposée utilise plutôt l'espace latent appris lors de la précédente phase de construction des HI, pour prédire directement le RUL avec des réseaux neuronaux récurrents et probabilistes. Enfin, une nouvelle proposition d'utilisation des trajectoires de HI est développée pour l'identification du mode de défaillance. Cette proposition prospective utilise une méthode existante de clustering de séries temporelles univariées, à savoir le clustering agglomératif avec comme métrique de distance le dynamic time warping (DTW). Ces trois propositions d'utilisation de l'espace latent ou des trajectoires de HI montrent des résultats prometteurs ouvrant la voie à de futures perspectives.

Abstract

With the rapid advancements in sensing technologies and the development of robust data storage infrastructure, the continuous acquisition of monitoring data has become feasible. These data mainly result as collections of multivariate time series (MTS) which feature several variables that exhibit temporal variations. The accessibility to these MTS monitored over long periods of complex systems' life present valuable opportunities to gain profound insights into these systems' underlying dynamics. In engineering, the field of prognostic and health management (PHM) can benefit from having access to such MTS. PHM aims to ensure the satisfactory operational conditions of physical systems throughout their lifespan. MTS issued from the monitoring of the evolution of these systems' signals can indeed help at providing robust prediction regarding their future states. In this thesis, the specific tasks of PHM related to MTS learning are studied. The methods of machine learning and inferential statistics applied to MTS are explored to solve these PHM tasks. A focus on deep learning and neural networks is applied due to their recent advancements and performances in various domains. Among the four tasks of PHM, namely, anomaly detection, remaining useful life (RUL) prognostic, failure mode identification and health assessment, the latter is studied in more details in the present thesis. It involves the construction of a health indicator (HI), i.e. a quantify of interest evolving with time and indicating the current degradation status of the system. It is indeed argued in this work that this task is central to PHM and can help at solving the other more specific tasks. A novel similarity-based HI learning method using similarity learning with a siamese neural network and a new contrastive loss is proposed. This method is specifically designed to enforce the monotonicity of the resulting HI which is a crucial property this latter must satisfy. This method is tested on two public datasets providing monitored signals on turbofans and rolling bearings respectively. These experiments validate the design of our method for enforcing monotonicity and shows improvement compared to existing works. The proposed method thus provides with monotonous HI trajectories, but also a learnt latent space where samples of the monitored signals from the system can be projected. This latent space is low-dimensional, and typically represents samples that share similar levels of degradation close together. Based on this novel HI learning approach, two methods are proposed for predicting the RUL, i.e. the time left for an instance of the system under study to reach failure. One method is based on the forecasting of partial HI trajectories coupled with a failure threshold estimation, both of which are carried out in an integrated approach combining stochastic process modelling and neural network approaches. The second proposed method rather uses the learnt latent space of the previously developed HI learning method, to directly predict the RUL with probabilistic recurrent neural networks. Finally, a new proposed usage of HI trajectories is developed for achieving failure mode identification, another important PHM task. This prospective proposition makes use of an existing univariate time series clustering method, namely agglomerative clustering with dynamic time warping as a distance metric. These three proposals of using either the latent space or the HI trajectories of the proposed HI learning model, show promising results and open for discussions and future reachable perspectives.

Remerciements

Ce manuscrit documente les travaux de recherche que j'ai effectué durant ma thèse qui aura duré un peu plus de trois années. Je me dois désormais de remercier ici toutes les personnes qui ont contribué à leurs manières au succès de cette thèse de doctorat.

Tout d'abord, je remercie tous les membres du jury: Engelbert Mephu-Nguifo, Zeina Al Masry, Bruno Sudret et en particulier les rapporteurs Piero Baraldi et Christian Gogu pour avoir pris le temps d'évaluer mes travaux, et d'y avoir apporté de précieuses critiques, ajustements et perspectives, que cela soit dans leurs rapports, pour les rapporteurs, ou lors de la soutenance pour l'ensemble des membres du jury.

Je tiens également à remercier très chaleureusement mon directeur de thèse Jean-Marc Bourinet et ma co-encadrante Cécile Mattrand pour leur accompagnement lors de ces trois années. Que cela soit pour leurs disponibilités, leurs apports méthodologiques et organisationnels, leurs expertises scientifiques, leurs relectures pour les articles et ce présent manuscrit ainsi que leur soutien moral lors des phases les plus critiques du doctorat, tous ces différents apports qui furent d'une grande aide pour parvenir avec succès au terme de ce doctorat. Je les remercie également de m'avoir permis d'intervenir lors de travaux dirigés et travaux pratiques dans leurs cours de statistiques et probabilités ainsi que d'apprentissage automatique et apprentissage profond.

Je remercie aussi à cet égard Katyanne Farias de m'avoir permis d'encadrer ses travaux dirigés et travaux pratiques en algorithmique et programmation lors de ma deuxième année.

Je remercie Francesco Canceliere, doctorant, pour m'avoir initié aux méthodes des filtres particulaires et avec qui j'ai eu la chance de collaborer sur ce sujet, une collaboration qui a mené à une communication écrite et orale lors d'une conférence internationale à Bologne en novembre 2023.

Je remercie tout le personnel administratif et technique de SIGMA Clermont, UCA et EDSPI pour leurs aides quotidiennes et sans qui aucune activité de recherche pourrait avoir lieu, un remerciement particulier à celles et ceux avec qui j'ai eu l'occasion d'échanger : Jacqueline Mabedene, Franziska Fisher, Sabine Sadargues, Virginie Mendes, Olivier Bullat, Véronique Sébert.

Je remercie également tous mes collègues doctorants pour le soutien mutuel et les moments conviviaux qu'on a pu partager qui sont aussi d'une importance cruciale dans la réussite de mon doctorat.

Finalement, je remercie mes proches, amis, colocataires et ma famille pour avoir su me soutenir dans les moments les plus difficiles et profiter avec moi dans les moments les plus gratifiants de ces trois années de thèses.

Contents

I

Introduction

I–1 General context

With the rapid advancements in sensing technologies and the development of robust data storage infrastructure, the continuous acquisition of monitoring data has become feasible. These data are predominantly gathered from sensors, but may also involve human interventions. They mainly result as collections of multivariate time series (MTS) which feature several variables that exhibit temporal variations.

The accessibility to these MTS monitored over the lifespan of a complex system presents valuable opportunities to gain profound insights into its underlying dynamics. It also enhances our ability to infer past, present, and future states of the system.

Numerous scientific domains focus on the analysis of MTS as a central aspect of their research. Earth system sciences, including climatology, meteorology, hydrology, geology, seismology, and vulcanology utilise such data, see e.g [43, 193, 228]. Environmental sciences employ MTS to investigate species population dynamics in ecology, see e.g. [157, 182, 182, 126, 48, 147, 73]. Health sciences leverage them to study complex patterns, such as in electroencephalogram (EEG) recordings of patients [128, 130, 24]. Even social sciences, such as long-term sociological and economic studies conducted on population scale, employ MTS analysis [72, 119, 134, 17]. In the present work, the domain of application is rooted in engineering sciences, which also heavily relies on the analysis of MTS, see e.g. [16, 35, 12]. In engineering, a primary objective for managing complex physical systems is to ensure their satisfactory operational conditions throughout their lifespan. To achieve this, it is essential to monitor the evolution of the system's health status and provide robust prognoses regarding future states. This field of engineering science is known as prognosis and health management (PHM) [159, 96, 236].

PHM involves the analysis of multiple quantities of interest (QoIs), which are monitored throughout the life of the system and manifest themselves as MTS. These QoIs are captured by various sensors, generating a significant amount of data. Consequently, specialised techniques and tools are required to extract relevant information from these MTS, which possess specific characteristics. MTS have potential length variations and irregular sampling frequencies. They share information between variables, that encapsulate the behaviour of the system under study, which makes the different MTS variables potentially interdependent. In addition, each variable of these MTS may exhibit different properties such as stationarity, linearity or Gaussianity, and inversely, non-stationarity, non-linearity or non-Gaussianity. These properties require a special attention in the modelling of the MTS and developed analysis methods, which are at the core of the field of MTS learning.

I–1.1 Machine learning

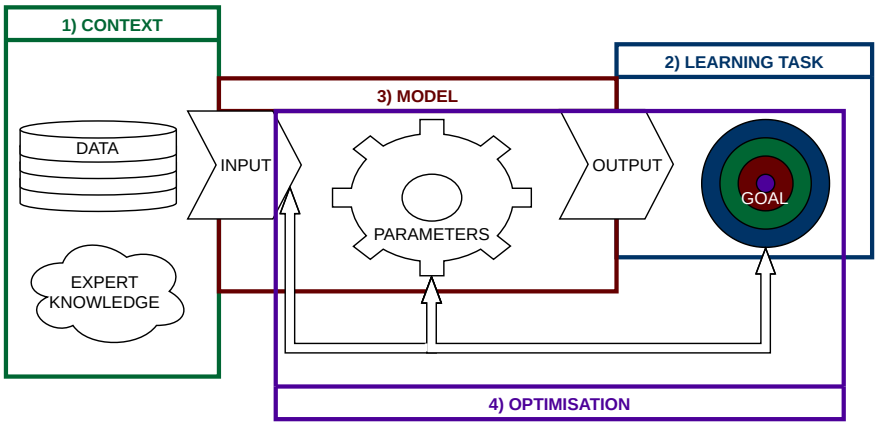


Figure I.1: Diagram for visualising the four elements of machine learning

In the field of data science, learning refers to specific algorithms that aim to produce desired outcomes based on data and statistical tools, without humans explicitly programming how to produce the results. Therefore, it can be seen as if the "machine" is learning by itself, giving rise to the term machine learning. To realise such an algorithm, the amalgamation of four fundamental elements is needed, as shown in Figure I.1. The first element encompasses the context of application, which includes the data to be processed, as well as any a-priori knowledge pertaining to the context. This knowledge includes known relationships between different variables in the data, information about the data extraction process, and other relevant contextual information. The second element consists in the definition of the specific learning task that necessitates resolution, entailing the exact nature of the knowledge to be extracted by the model. The model itself constitutes the third element, and must be fully defined by a mathematical composition of operations with multiple parameters. It is designed in direct correlation with the objective at hand and the characteristics of the data it operates on. The fourth element is the optimisation procedure, which entails the search for the optimal parameters of the model to effectively perform the learning task as much as the model complexity allows for. While in numerous machine learning methods, model and optimisation are intertwined, it is essential to discern between the two for a comprehensive overview of the state-of-the-art of MTS learning.
An introduction to the first element of machine learning, i.e. the context of application, for the scope of this thesis is necessary before diving into the three other elements. The next section therefore frames this context of application.

I–1.2 Application context: Prognostics and Health Management

Prognostics and health management (PHM) is a field that plays a key role in ensuring the reliability and optimal performance of complex systems. It encompasses a range of techniques and methodologies aimed at predicting the future health condition and potential failures of such systems [184]. The overarching goal of PHM is to enable timely decision-making for maintenance and asset management, ultimately enhancing system availability, reliability, sustainability and reducing operational costs. PHM involves three distinct stages: data acquisition, data analysis, and decision-making

[236] which are depicted in Figure I.2.

In the data acquisition stage, relevant sensor data and information about the system under consid-

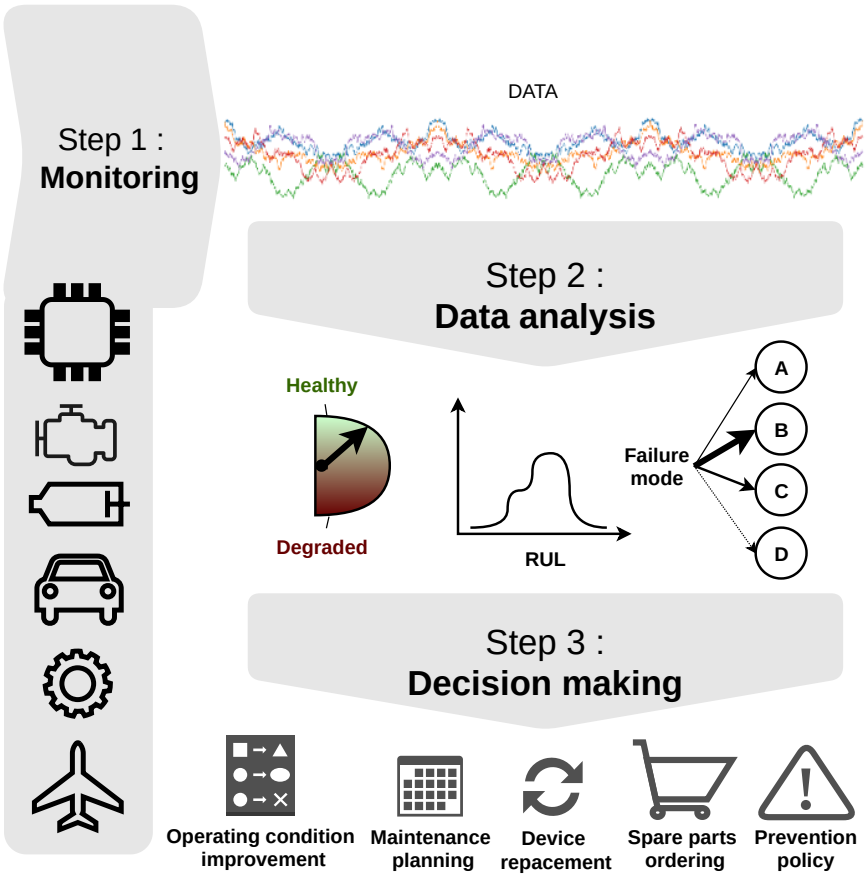


Figure I.2: Diagram illustrating the three steps of PHM

eration are collected. Such data can be of various types, such as vibrations, temperatures, pressures, fluid flows or other measurable parameters, depending on the specific application domain [144]. The data acquisition stage requires careful planning to ensure accurate and reliable measurements, as well as the selection of appropriate sensors.

Once the data has been acquired, the data analysis stage begins. This stage involves processing, modelling, and extracting meaningful insights from the collected data. Various fields of mathematics and computer science, such as signal processing, statistical analysis and machine learning, are employed to uncover patterns, trends, and anomalies in the data. The data analysis stage e.g. aims to identify early indicators of potential failures or deterioration in the health of the system. This is where MTS learning, the focus of our research, comes into play [236].

The final stage of PHM is decision-making. Based on the analysis and insights gained from the data, decisions are made regarding maintenance actions, repairs, component replacements, or other appropriate interventions. The goal is to optimise maintenance schedules, minimise downtimes, and maximise the operational lifespan of the system. Decision-making in PHM often involves a combination of human expertise and automated decision support systems that integrates the findings from data analysis with domain knowledge and operational requirements.

In this manuscript, we will solely focus on the data analysis stage within PHM, leveraging MTS learning techniques to address key challenges and tasks. However, understanding the context of data acquisition and decision-making is crucial, as it provides a holistic perspective and facilitates

the development of effective data analysis methodologies. Subsequently, we will delve into the main data analysis tasks within PHM, namely anomaly detection, failure mode identification, health assessment, and remaining useful life (RUL) prognostics. The following sections try to provide comprehensive explanations of these four essential tasks.

I–1.2.1 Anomaly detection

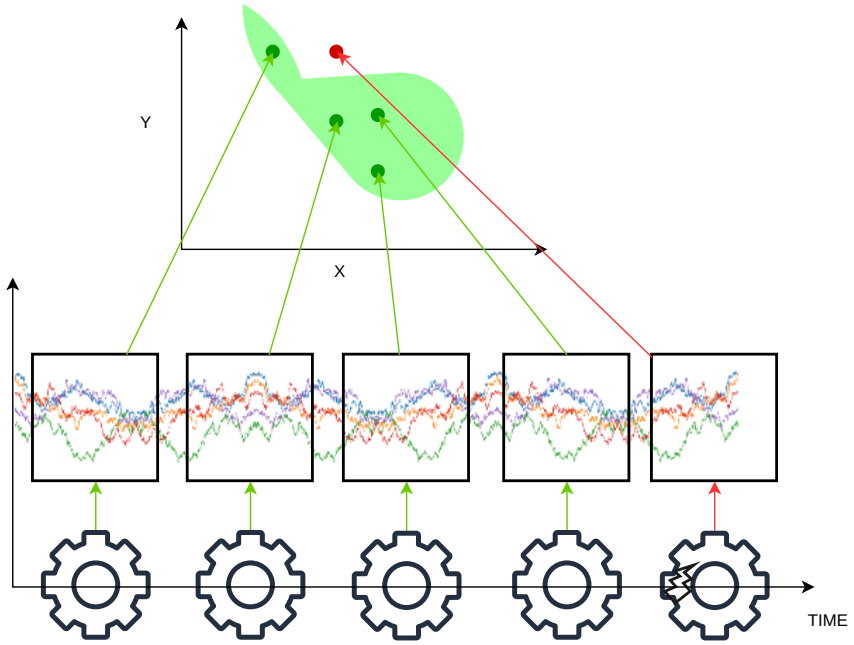


Figure I.3: Illustration of anomaly detection task in PHM

Anomalies refer to deviations from the expected behaviour of a system or its components. Anomaly detection plays a crucial role in identifying abnormal patterns or events that may indicate the presence of faults, malfunctions, or potential failures [11]. It enables early detections of emerging faults or degradations in the system's health, making proactive maintenance interventions possible to return to normal condition and minimizing the risk of unexpected failures. Data being in the form of MTS, it is therefore the field of anomaly detection in time series that is put to contribution. Anomaly detection can be formulated in two different machine learning tasks, as follows.
If enough data samples are available for both normal and abnormal conditions, and that each of these data samples is appropriately labelled normal or abnormal, then the formulation of the anomaly detection task takes the form of classification, i.e. a supervised setting.
However, in most situations, and especially with complex systems, it is challenging to label data as normal or abnormal. The task is therefore often formulated in an unsupervised setting where only data samples in normal condition are available [5]. The task then consists in learning the distribution of the normal data samples. A sample that deviates significantly from this distribution is then considered as an anomaly. With high dimensional data such as multivariate time series, learning distribution is not feasible, so dimension reduction or representation learning are used to detect anomaly in lower dimension spaces. An illustration of this idea is depicted in Figure I.3

I–1.2.2 Failure mode identification

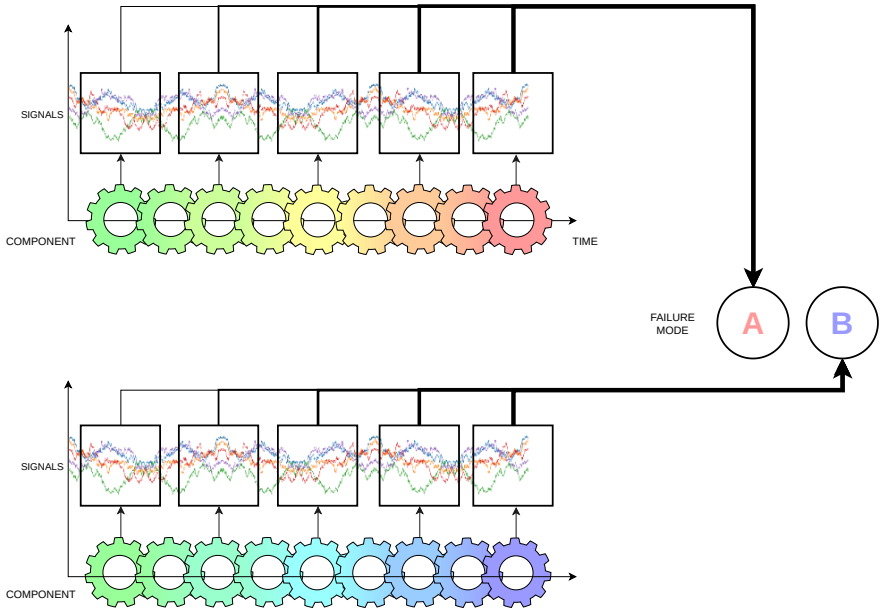


Figure I.4: Illustration of failure mode identification task in PHM

Failure mode identification is a critical task aimed at understanding the specific modes or types of failures that a system or its components may exhibit, see Figure I.4 for a graphical illustration. It involves the categorisation and characterisation of these different failure modes, which can arise due to various factors such as, e.g. ageing, wear or tear, and that are determined by environmental and operational conditions. Failure mode identification provides valuable insights into the root causes and mechanisms of failures, enabling targeted maintenance and mitigation strategies. The field of MTS learning contributes to this task by formalising it as an MTS classification (supervised) or clustering (unsupervised) problem, depending on whether the data samples of the context are labelled with the appropriate failure mode or not [69]. These two different settings make two different end goals, where one asks for each new sample to assign it to a predefined failure mode, while the second must first separate existing samples into different modes and then assign new samples to these modes. The second setting is interesting for one that has wishes to discover potential failure modes for a complex system with little information on the internal degradation dynamics.

I–1.2.3 Health assessment

Health assessment plays a crucial role in evaluating the overall condition and performance of a system or of its components. It involves the quantitative description of a health indicator to assess the system's operational status, potential degradations, or deviations from the desired performance [98], as shown in Figure I.5. As opposed to the previous tasks, health assessment aims to provide a holistic view of the system's health and identify any signs of deterioration or abnormalities that may impact its reliability or functionality. Health assessment enables maintenance decision-making, including determining the need for maintenance actions, scheduling inspections or initiating repairs. The purpose of health assessment is to identify the position of the system on some degradation trajectories. It can be formalised as the development of a health indicator (HI) whose values are

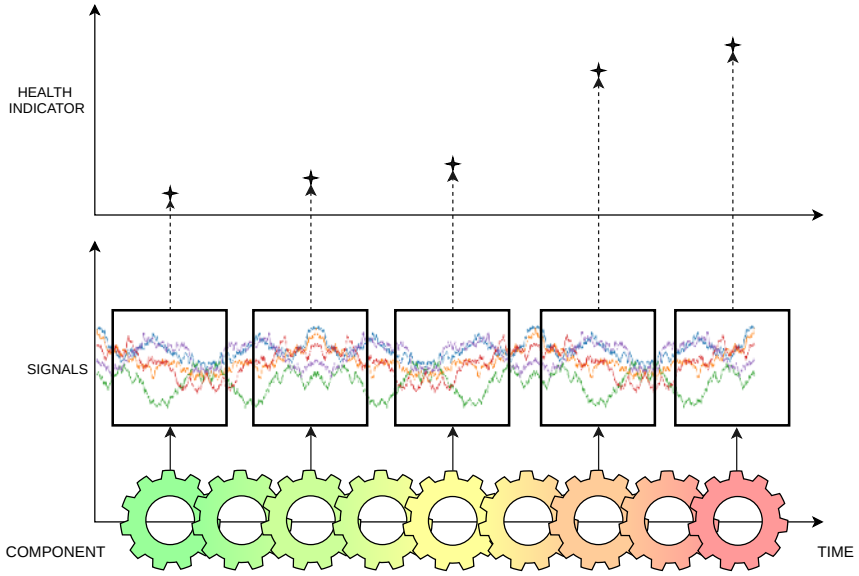


Figure I.5: Illustration of health assessment task in PHM

assessed at regular time steps. In terms of MTS learning, it corresponds to a task of extrinsic regression, i.e. determining the value of a quantity of interest that is not measured (the HI value) via other quantities of interest that are measured (the data sample). It can be defined in a supervised setting if the HI values are available for enough data samples, alternatively the HI label can be set via hypotheses, i.e. linear evolution from 0 to 1. Unsupervised settings similar to the ones used for anomaly detection are also possible. The operation then consists in reducing the dimension of the data samples in a lower dimensional space and evaluating the HI as the distance between the normal condition data sample distribution and the data sample to be tested.

I–1.2.4 RUL prognostics

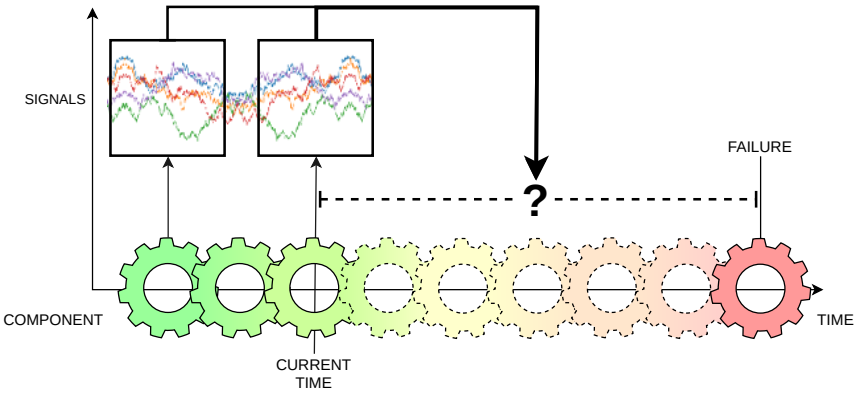


Figure I.6: Illustration of RUL prognostics task in PHM

RUL prognostics aims to estimate the remaining operational lifespan of a system or of its components. It involves the prediction of the time remaining until a failure or a predefined performance threshold is reached. Figure I.6 gives an illustration of this task. RUL prognostics is crucial for

effective maintenance planning, as it enables timely interventions to prevent unexpected failures, optimise maintenance schedules and maximise the utilisation of system assets. RUL prognostics can take into account various factors, including operating conditions, maintenance history and the severity of degradation. This task can be formalised in two different ways. First, in the continuation of the health assessment task, it can be an intrinsic regression task, and more precisely the forecasting of a HI until it reaches a predefined failure threshold. Secondly, skipping the health assessment task, it can also be represented as an extrinsic regression, i.e. predicting an unknown quantity (remaining time until failure) based on other quantities (MTS data samples).

I–2 Motivations and objectives

The four tasks of the PHM data analysis phase, presented above, have different end goals. However, one of them seems to be connected to all the other tasks. Indeed, health assessment, which enables to continuously assess an indicator of health, is on some different aspects linked to the other tasks. First, it is a more complete task than anomaly detection. While the latter is only about detecting an abnormal behaviour, heath assessment proposes to quantify the deviation from the normal behaviour. Therefore, anomaly detection can be performed from the knowledge of a HI. Secondly, as mentioned above, the task of RUL prognostics can also be formulated as the forecasting of a HI, therefore the health assessment task is here directly needed. Finally, the failure mode identification can be based on the entire historical evolution of a component and, in that sense, a HI trajectory is a perfect representation of the historical measurements in a reduced dimensional space. One can then imagine failure mode identification as a classification or clustering task based on one or multiple HI trajectories.
Based on this analysis, the core of this PhD thesis is to explore the analysis of PHM data focusing on health assessment. The main objective is then to thoroughly explore the health assessment task and propose a new generic method for it. Additionally, this work explores the possibility of using the HI resulting from health assessment for the other tasks of failure mode identification, RUL prognostics and anomaly detection.
To achieve the mentioned objectives, a thorough investigation of the literature on the subject is realised, centred around these two research statements (RS):

RS1 What are the current limitations of current health assessment methodologies?

RS2 How does the PHM literature relate health assessment to other tasks in PHM?

Based on the results of this literature exploration, a new generic method for health assessment that attempts to overcome the limitations identified via **RS1** is proposed. And, based on the outcomes of **RS2**, this work proposes a new usage of HI for failure mode identification and improvements in the usage of HI for RUL prognostics.

I–3 Structure

After a brief reminder of the main theoretical and practical concepts of machine learning and MTS, a thorough review of the literature aiming at answering the two research statements is outlined. The present work then proposes solutions to contribute to the field of prognosis and health management in the identified research directions.

More precisely, the following parts of this thesis manuscript are organised as follows:

Chapter II presents some elements of theory and practice for machine learning, especially for handling MTS data. The main types of learning tasks on MTS are discussed, as well as the most common and successful learning methodologies for solving them, including modelling practices and optimisation procedures.

Chapter III introduces the review of the literature oriented on the two research statements presented in Section I–2. It first explores the expected properties of a useful health indicator. Then, a review of the different health indicator construction methods is presented as a categorisation of different approaches, with examples for each category. Each approach is discussed in relation to its ability to provide HIs compliant with the expected properties. It is found that a particularly important property, i.e. monotonicity, is acknowledged to be crucial while almost never constrained in most existing approaches. Finally, a review of the different uses of HI for the other PHM tasks is presented. The task of RUL prognostics is unsurprisingly the most studied in this direction, as the HI concept was firstly designed to perform it. The other tasks, however, seldom relies on HI construction. Nevertheless, it is found that HI construction induces to extract general features from the data and that these features as well as the obtained HI could be used to solve the other tasks of anomaly detection and failure mode identification.

Chapter IV presents our proposed approach for HI learning. It undertakes to tackle the main current limitation identified in the preceding review. The monotonicity of the HI is integrated as a constraint directly in the model optimisation, without any additional hypothesis on the particular form of the evolution of the HI. Moreover, the approach is framed in a contrastive self-supervised setting aimed at finding a representation space where distances can be seen as distances in terms of health, and therefore used as HI. This framing involves the projection of data samples in a representation space composed of learned features related to the system degradations.

Chapter V explores the idea of using the HI curves learned previously to solve other tasks of PHM than health assessment. It first focus on exploring an often dismissed possibility of using them for identifying failure modes with classical uni-variate time series clustering techniques. Then this chapter explores the possible usage of the HI trajectories and the representation space learned in the previous chapter for RUL prediction. A first method based on the combination of particle filtering and neural networks is proposed for this task. Then a second approach based on self-supervised learning and probabilistic neural networks is explored.

Chapter VI Concludes the thesis and open some interesting perspectives.

I–4 Publications and communications

Journal articles :

- Engineering Applications of Artificial Intelligence: **Similarity learning for predictive maintenance: health indicator construction based on siamese neural networks and contrastive loss**, Etienne Jules, Cécile Mattrand and Jean-Marc Bourinet, **(under review)**

International conference publications :

- Proceedings of 2023 7th ICSRS: **Remaining useful life prediction of turbofans with virtual health indicator: a comparison of particle filter-based approaches**, *Etienne Jules, Francesco Cancelliere, Cécile Mattrand and Jean-Marc Bourinet* **(waiting for publication)**

International conferences - oral communications :

- ECCOMAS, June 2022, Oslo: **Health indicator learning for predictive maintenance based on a triplet loss and deep siamese network**, *Etienne Jules, Cécile Mattrand and Jean-Marc Bourinet*
- ICRE, November 2023, Bologna: **Remaining useful life prediction of turbofans with virtual health indicator: a comparison of particle filter-based approaches**, *Etienne Jules, Francesco Cancelliere, Cécile Mattrand and Jean-Marc Bourinet*

Seminars :

- INPxIAE, January 2023, **Estimation de l'état de santé de roulements à partir de séries temporelles**, *Etienne Jules*
- Journée scientifique des doctorants de l'EDSPI 2021: **Learning multivariate time series**, *Etienne Jules, Cécile Mattrand and Jean-Marc Bourinet*

Posters :

- Journée scientifique des doctorants de l'EDSPI 2021: **Learning multivariate time series**, *Etienne Jules, Cécile Mattrand and Jean-Marc Bourinet* **(best poster award)**

II

Machine learning and multivariate time series: theoretical and practical considerations

II–1 Time series and their acquisition in PHM

II–1.1 Time series

A **time series**, also named signal, can be defined as an ordered set of values, or data points, indexed by time, i.e. each data point represents the value at a specific time of one, or multiple, quantities of interest. If the data points of such a time series are single values, then it is called a univariate time series (UTS), while if the elements are vectors of several values, it is referred to as a multivariate time series (MTS). Because time series are actual monitored measurements, they cannot be continuous in the strict sense and are therefore discrete in nature, although they can be modelled, i.e. interpolated or regressed, by continuous functions.

A time series can be viewed as a realisation of a discrete-time or continuous-time **stochastic process**. It can also be referred to as a trajectory. A stochastic process may be introduced as a collection of random variables (univariate case) or vectors (multivariate case) $\{X_t, t \in T\}$ indexed by a parameter t varying in an index set (finite or not) T:

$$\begin{aligned} X : T &\to \mathcal{V}(\Omega, \mathcal{F}, \mathbb{P}, E) \\ t &\to X_t : \Omega \to E \\ \omega &\to X_t(\omega) \end{aligned}$$

Where $\mathcal{V}(\Omega, \mathcal{F}, \mathbb{P}, E)$ denotes for the vector space of E-valued random variables, or vectors, defined on the probability space $(\Omega, \mathcal{F}, \mathbb{P})$. If $E = \mathbb{R}$, or $\mathbb{R}^d$ where d is the number of variables, the stochastic process is said real valued.
For a fixed $t \in T$, X_t is a random variable and $X_t(\omega)$, $\omega \in \Omega$ is a realisation of such a variable.
Another possible definition of a stochastic process is given by:

$$\begin{aligned} X : (\Omega, \mathcal{F}, \mathbb{P}) &\to E^T \\ \omega &\to X_\omega : T \to E \\ t &\to X_\omega(t) \end{aligned}$$

With such a definition, for a given outcome $\omega \in \Omega$, X_ω is a function of time and is called a trajectory, or sample path. A sample of discrete values along this trajectory corresponds to a time series. A visual interpretation of a real-valued stochastic process is given in Figure II.1

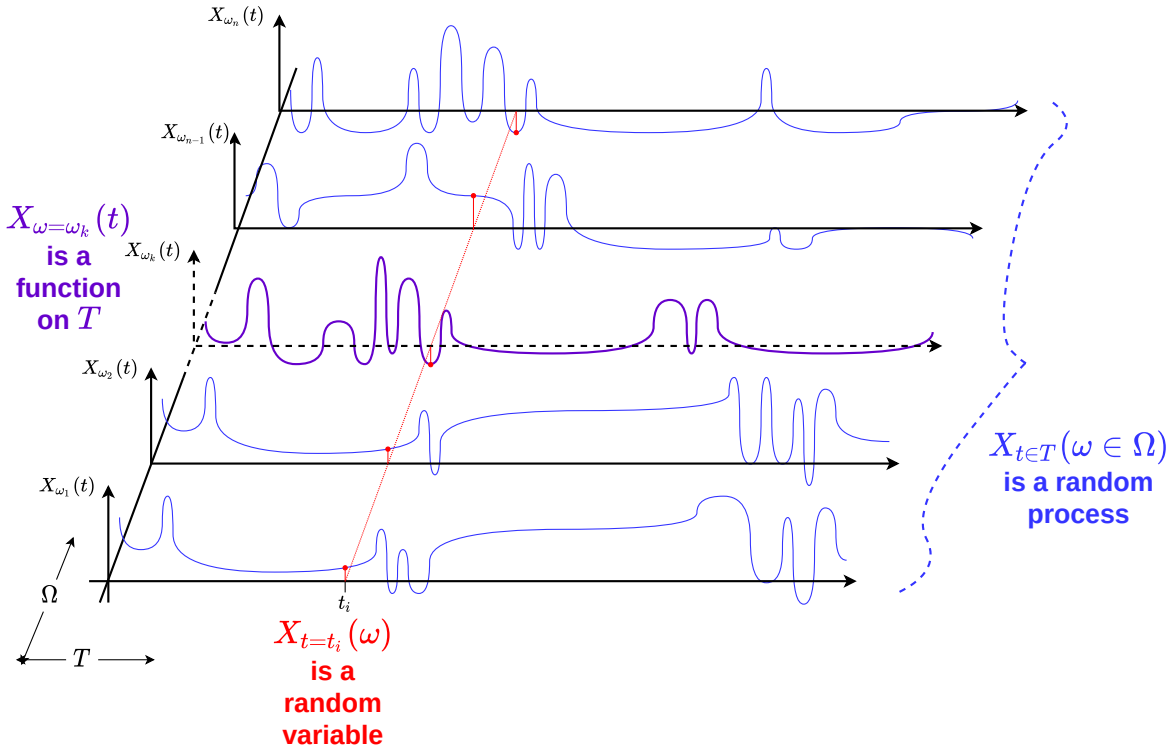


Figure II.1: Illustration of a stochastic process when considering T as a representation of time.

An important property of stochastic processes is **stationarity**. A stochastic process is said to be strictly stationary if its joint probability distribution is invariant in time, that is:

$$\forall t_1, t_2, \ldots, t_n, \tau \in T \text{ with } \tau > 0 \text{ and } 0 \leq t_1 \leq t_2 \leq \cdots \leq t_n :$$
$$F_{X_{t_1}, X_{t_2}, \ldots, X_{t_n}}(x_{t_1}, x_{t_2}, \ldots, x_{t_n}) = F_{X_{t_1+\tau}, X_{t_2+\tau}, \ldots, X_{t_n+\tau}}(x_{t_1+\tau}, x_{t_2+\tau}, \ldots, x_{t_n+\tau})$$

where $F_{X_{t_1}, X_{t_2}, \ldots, X_{t_n}}(x_{t_1}, x_{t_2}, \ldots, x_{t_n}) = \mathbb{P}(X_{t_1} \leq x_{t_1}, X_{t_2} \leq x_{t_2}, \ldots, X_{t_n} \leq x_{t_n})$ denotes for the joint cumulative distribution of the subset of variables $\{X_{t_1}, \ldots, X_{t_n}\}$.
This property can be challenging to verify in practice, if at all possible, so it is often sufficient to consider wide sense stationarity. A stochastic process is said to be wide-sense stationary if its first and second-order moments, i.e. mean and autocovariance are constant through time. This may be formalised as follows:

$$\forall t \in T,\ \mu_X(t) = E[X_t(\omega)] = \mu_X$$
$$\forall t, t' \in T,\ \gamma_X(t, t') = E[(X_t(\omega) - \mu_x)(X'_t(\omega) - \mu_x)] = \gamma_X$$

Time series inherit the property of their underlying stochastic process.es Stationary stochastic processes, and by extensions stationary time series, have been largely described and studied in the statistic's literature. However, in practice, in the real world, one often has to manipulate non-stationary time series and can therefore not blindly rely on classical stationary stochastic process theories and tools.

Additionally, when considering real case scenarios, one often has access to multiple signals issued from the same system and thus mutually dependent. With MTS, another difficulty thus arises in capturing potential **dependences between variables**. This greatly increases the complexity of precise and grounded statistical models needed for understanding the behaviours of such multivariate stochastic processes.

These two challenges, **non-stationarity** and **multi-variability** of real case time series, must then be taken into consideration. Classical stochastic process theory might be underdeveloped at the moment to fully model such time series, therefore incorporating the use of deep learning theories and tools for solving tasks involving MTS has become of particular interest and is a favoured research direction in this work.

II–1.2 Context of time series acquisition in PHM

(a) Context of PHM

As mentioned in the introductory chapter, in PHM, when the phase of signal analysis has arrived, knowing the context of acquisition of the data, i.e. the signals or time series, is of prime importance. According to a vast number of works on the subject, most of the challenges in the PHM data analysis phase are actually a direct consequence of the monitoring phase [190, 98, 236]. More precisely, in a recent paper attempting to compile an overview of PHM development [236], the current challenges encountered are summarised. Among them, four challenges are directly linked to the monitoring phase of PHM: missing and anomalous data, lack of labelled data, data scarcity and high variability of unmeasured operational and environmental condition during the monitoring process. These mentioned challenges are all determinant for improving the PHM development and must be taken into consideration in the monitoring stage, but they are also challenges to which the data analysis phase must adapt to. It is therefore crucial to understand the context of monitoring before diving into the data analysis phase.

In this thesis, the core of our research is nevertheless not to adapt to all the potential defect in the monitoring phase. More precisely, the available MTS are considered complete, i.e. with no missing values or anomalies. The data scarcity, lack of labelled data and variability of unmeasured operational and environmental condition are also not at the core of the thesis, and thus not prioritised in the development of proposed solutions. Nevertheless, in the concrete application case studied in the thesis, these challenges do arise, they can be considered as missing knowledge on the data and therefore treated as uncertainties. Accounting for uncertainties is therefore another challenge that this work attempt to face, which related to the field of **uncertainty quantification**.

The last important consideration on MTS data treated in this work is the following: the signals are considered as being synchronised and regularly sampled. The synchronisation reflect the fact that every variable of a MTS are monitored at the same time instants. The hypothesis of regular sampling is detailed in the next paragraph, which introduces considerations on the different sampling types.

(b) Sampling types

Let first consider MTS acquired from multiple instances of a studied system, denoted $\mathcal{S}$. These instances are considered to be run until failure, and are thus also called run-to-failures (RtF). The number of recorded instances of the system is hereafter denoted n and d is the number of monitored variables. An essential characteristic of time series, either UTS or MTS, is the duration between each of their measurements, i.e. their sampling. We here differentiate four classes of sampling types, see Figure II.2.

Uniform sampling Here, the time duration between two consecutive measurement is consistent throughout the series with a sampling frequency $f_s = \frac{1}{T_s}$. With such a sampling, the MTS acquired from n instances can be defined as follows:

$$\mathcal{S} = \{s^{(k)}(t)\}_{k\in\mathcal{K},\, t\in\mathcal{T}_k(T^{(k)})}$$
$$\text{with } \mathcal{K} = \{1,\dots,n\};\ \mathcal{T}_k(T^{(k)}) = \{0, T_s, \dots, T^{(k)}\}$$

where $s^{(k)}(t) \in \mathbb{R}^d$ are the values recorded for the d signals at time t and where $T^{(k)}$ is the time of failure of the k^{th} instance of the studied system.

Subseries-uniform sampling Let first define a subseries as a shorter series included in a longer one which contains all the elements of the latter between two given time instants. Given this definition, a time series can be sampled irregularly on a global scale but uniformly considering a smaller scale, i.e. when regarding subseries. More precisely, a sampling can be considered subseries-uniform if there are two time deltas — Δ_L the subseries length and Δ_G the gap between subseries starting times — and a sampling frequency f_s, such that every subseries of size Δ_L, starting from the one at $t = 0$ and each spaced by Δ_G, are uniform with sampling frequency f_s. See Figure II.2 for a visual interpretation. With such a sampling, the MTS acquired from n instances can be defined as follows:

$$\mathcal{S} = \{s^{(k)(i)}(t)\}_{k\in\mathcal{K},\, i\in\mathcal{I}_k(T^{(k)}),\, t\in\mathcal{T}_{ki}}$$
$$\text{with } \mathcal{K} = \{1,\dots,n\};\ \mathcal{I}_k(T^{(k)}) = \{0,\dots,I^{(k)}\};$$
$$\mathcal{T}_{ki} = \{i\Delta_G, i\Delta_G + T_s, i\Delta_G + 2T_s, \dots, i\Delta_G + \Delta_L - T_s, i\Delta_G + \Delta_L\}$$

With $I^{(k)} = \frac{T^{(k)} - \Delta_L}{\Delta_G}$.

Subseries-semi-uniform sampling This sampling type is similar to the subseries-uniform sampling except that Δ_G, the gap between subseries starting time is not consistent throughout the time series. No notations are introduced for time series with such a sampling. This type of sampling is indeed not considered in this work. However, in cases where the irregularity in precise time deltas Δ_G is marginal or compensated by a regularity in operation cycles, they can be considered as subseries-uniform. For instance, monitoring the ten first seconds of electrical signals of batteries for each charging cycle would lead to time gaps between each subseries of ten seconds to not be regular, but it can still be considered as regular in terms of operation cycle.

Irregular sampling If it belongs to none of the three previous cases, a monitoring is considered as irregular. However, similarly as for the previous case, if a time series is monitored such that T_s is not constant throughout the series but that the irregularity of T_s are marginal

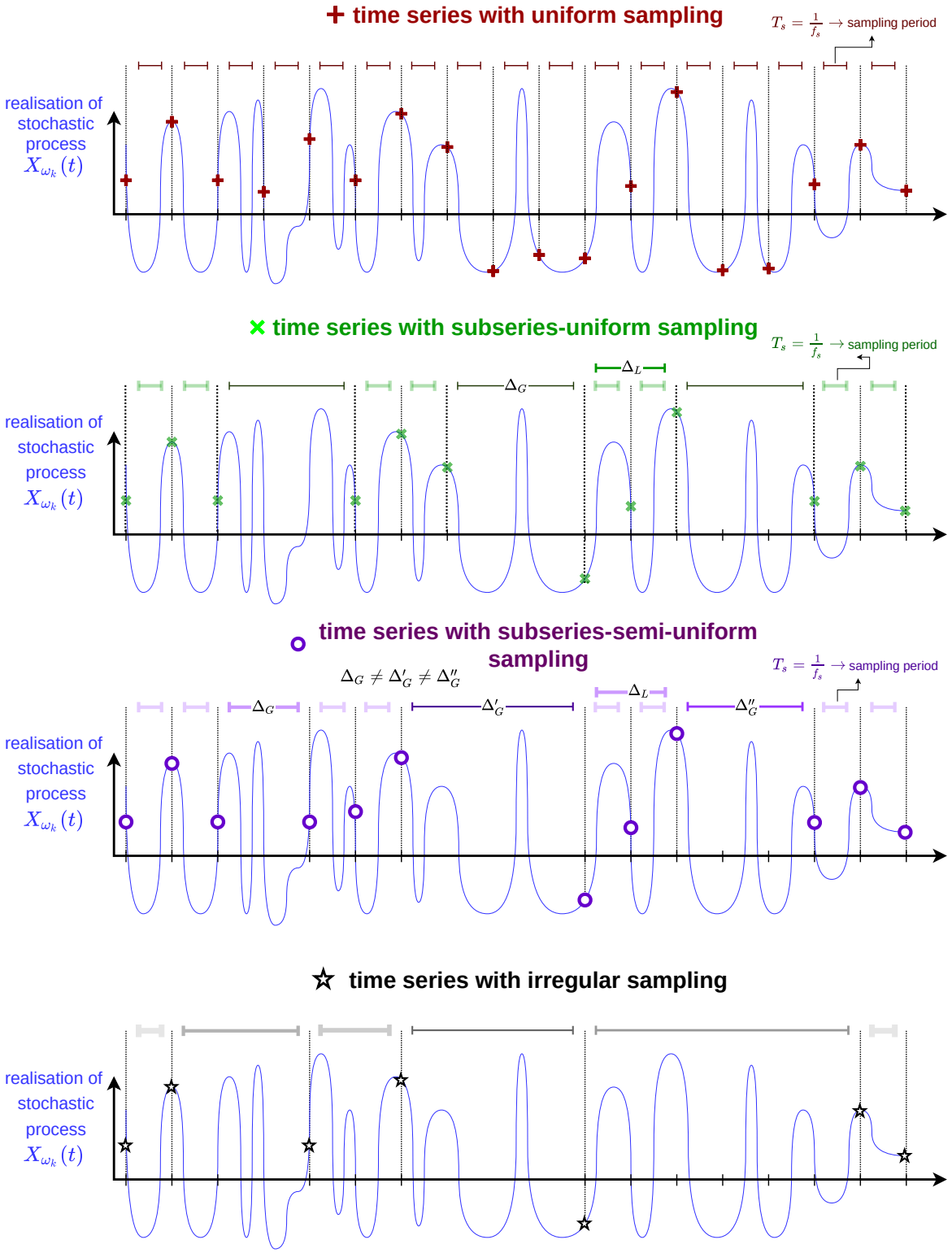


Figure II.2: Time series with different sampling types on a continuous stochastic process realisation.

or compensated by a regularity in operation cycle, the sampling can be considered uniform.

For instance, monitoring specific signals of a aircraft's engine after each take-off would lead to different time duration between each measurement, but could be considered a uniform sampling in terms of operation cycle.

The sampling type of MTS is directly determined by the monitoring strategy, which can generally be either continuous or event-based, i.e. where measurements are obtained when triggered by special events. This latter strategy can lead to irregular or subseries-semi-uniform time series, while continuous monitoring lead to uniform or subseries-uniform time series.

We acknowledge the existence of solutions for learning on MTS that are not synchronised or irregularly sampled, e.g. reservoir computing [214, 160, 106, 236, 49], but it is out of the scope of this thesis. This latter is restricted to MTS with synchronised and uniform or subseries uniform sampling.

To resume, this section aims at giving an overview on the type of data, MTS, and their considered context of acquisition in the present work. One important idea that emerges is that these MTS are often **scarce**, i.e. there is often a low number n of instances, or RtF, for which a MTS is available. These MTS also depend on unknown variables and unmeasured operating and environmental conditions. This lack of information necessarily induces that the model processing these data will produce output that are also imprecise. These imprecisions must therefore be taken into account by quantifying the **uncertainties** of the outputs produced by models processing these MTS. Another important idea that must be put forward is that MTS monitored from real-world systems are often **non-stationary**. It is especially true in PHM, whose goal is to actually study degradations of the system, and to detect changes in the system behaviour, i.e. non-stationarities. Finally, the different variables of a MTS can be **mutually dependent** and also **depend on the previous values** of the MTS. There are, therefore, three major difficulties to take into account in this work which are the **non-stationarities** of the MTS, the **interdependence** of the multiple signals composing these MTS, and the **uncertainties** stemming from the low number of observed instances, the presence of unobserved variables and variable operating and environmental conditions.

II–2 Machine learning on time series

To extract valuable knowledge shared along the different variables of a MTS related to the characteristics and evolution of a system, special methods are required. These methods arise from inferential statistics and machine learning, and are discussed in this chapter. But, first and foremost, extracting information from data, e.g. learning, can be classified in different type of problems, and, when applied to MTS, this classification is further detailed.

II–2.1 General consideration on machine learning

In this section, the focus is on presenting the main concepts of machine learning in the general case of supervised learning. Supervised learning is a special case and other settings such as unsupervised, semi-supervised or self-supervised are important for this work, these latter will be defined later.

As mentioned in the introductory chapter, ML is the composition of four elements, starting with the **context**, that includes the data that need to be learned on, their meaning in the real world, and how they were obtained. This element is here represented by the dataset $\mathcal{D}$, composed of several examples or instances composed of numeric values of different variables:

$$\mathcal{D} = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y},\ 1 \leq i \leq N\} \tag{II.1}$$

Here (x_i, y_i) represent an example i, with x_i and y_i the numeric values of explanatory variables and explainable variables respectively. $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{Y} \subseteq \mathbb{R}^m$ are respectively the input and output space, which can also be called explanatory variables space and explainable variable space. N is the number of available examples.

This notation of the dataset is practical for supervised learning, but before another element of ML is introduced, i.e. the learning task, the separation of example variables into explanatory and explainable variables is purely artificial. This is precisely the role of the learning task to determine this separation. Indeed, in a supervised learning context, the learning task is always down to predict some explainable variables only with the knowledge of some explanatory variables. The precise nature of the learning task is actually determined by the definition of the explainable variable space $\mathcal{Y}$. If $\mathcal{Y} = \mathbb{R}$ then the learning task is a regression, if $\mathcal{Y} = \mathbb{R}^m$ then it is a multivariate regression. If $\mathcal{Y} = \{-1, 1\}$ the task is called binary classification and $\mathcal{Y} = \{1, 2, \ldots, e\}$ with $e > 2$ is the definition of multi-class classification.

Now that the context and learning task are defined, the model, the third element of ML as presented in Chapter I, can be introduced. The model is the mathematical object that is aimed at mapping the explanatory variable $\mathcal{X}$ to the explainable ones $\mathcal{Y}$. Consider the unknown following function:

$$f : \mathcal{X} \to \mathcal{Y} \tag{II.2}$$

$$x \mapsto y = f(x) \tag{II.3}$$

where x and y here denote for the explanatory and explainable variables of both available examples ($\mathcal{D}$) and unknown examples. The model can thus be defined as a functional or hypothesis space as follows:

$$\mathcal{F} = \left\{ \tilde{f} : \begin{array}{ccc} \mathcal{X} & \to & \mathcal{Y} \\ x & \mapsto & y = \tilde{f}(\cdot; \theta_{|\theta_h}, \theta_h) \end{array} \right\} \tag{II.4}$$

$$\text{s.t. } \exists\ \theta_h^* \in \Theta_h,\ \exists\ \theta_{|\theta_h^*}^* \in \Theta_{|\theta_h^*}\ ;\ \tilde{f}(\cdot; \theta_{|\theta_h^*}^*, \theta_h^*) \approx f(\cdot)$$

here $\theta_{|\theta_h}$ and θ_h respectively represent the parameters and the hyperparameters of the model $\tilde{f}$. Θ_h is the hyperparameter space and $\Theta_{|\theta_h}$ is the parameter space conditioned on θ_h. $\theta_{|\theta_h}^*$ and θ_h^* for their part represent the optimal values of $\theta_{|\theta_h}$ and θ_h for approximating f. The model is thus defined by

a hypothesis space $\mathcal{F}$, a set of functions $\tilde{f}$ that can be embodied by varying the values of parameters θ and hyperparameters θ_h. We here differentiate between parameters and hyperparameters. The difference between θ_h and $\theta_{|\theta_h}$ is that fixing the values for the former will only restrict the hypothesis space to a thinner space: $\mathcal{F}_{\theta_h} \subset \mathcal{F}$, while fixing the values for $\theta_{|\theta_h}$ is only possible if θ_h are already fixed, and directly restrict the hypothesis space to a single value $\tilde{f}(\cdot\ ; \theta_{|\theta_h}, \theta_h)$. To give a simple example, if the model is considered to be the set of all polynomial functions, then the order of the polynomial function is a hyperparameter, it only restricts the hypothesis space. The coefficients of the polynomial, however, when entirely set, completely define the polynomial function.
Now that these notions of parameters and hyperparameters is clarified, it becomes obvious that θ is conditioned on θ_h, emphasised by the notation $\theta_{|\theta_h}$ in Eq. (II.4).

Finally, the last element of ML now comes into play, which is the optimisation. It is by essence in the supervised setting an algorithm that aims at finding, for the model $\mathcal{F}$, the values $\widehat{\theta_h}$ and $\widehat{\theta}_{|\theta_h}$ which respectively approximate θ_h^* and $\theta^*_{|\theta_h}$ defined above. For this, the available data $\mathcal{D}$ is used for adjusting and assessing the fitness of successive parameters and hyperparameters values explored during the algorithm. Before the optimisation, the dataset $\mathcal{D}$ is typically divided in three subsets called training, validation and testing sets and respectively denoted as $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{test}}$. $\mathcal{D}_{\text{train}}$ is directly used to adjust the parameters, $\theta_{|\theta_h}$, given any hyperparameters values θ_h, during the optimisation process. $\mathcal{D}_{\text{val}}$ is used during the optimisation procedure to assess the quality of the parameters and hyperparameters combination and therefore helps in selecting the best hyperparameters. Finally, $\mathcal{D}_{\text{test}}$ is used at the end of the optimisation to simulate unknown data examples and therefore helps in estimating the generalisation capabilities of the model, i.e. to what extent the model is able to map the explanatory variables $\mathcal{X}$ to the explained variables $\mathcal{Y}$.
The optimisation is actually split in two different parts. A first optimisation algorithm $\mathcal{O}_1$ aims at finding, for the model $\mathcal{F}_{\theta_h}$, the best parameters $\widehat{\theta}_{|\theta_h}$ given θ_h and $\mathcal{D}_{train}$. Additionally, this optimisation procedure $\mathcal{O}_1$ often depends on its own hyperparameters defining its proper mechanism, they are here denoted as θ'_h. $\mathcal{O}_1$ typically defines a measure that quantifies how real values of the explainable variables $y = f(x)$ are close to the predicted values $\tilde{y} = \tilde{f}(x, \theta_{|\theta_h}, \theta_h)$. This measure is called the loss function ℓ and can be defined for a single example or a set of examples, respectively as follows:

$$\ell_i(\theta_{|\theta_h,\theta_h}) = \ell(y_i, \tilde{f}(x_i, \theta_{|\theta_h}, \theta_h))\ ,\ (x_i, y_i) \in \mathcal{D} \tag{II.5}$$

$$\ell_{\mathcal{D}_\mathcal{I}}(\theta_{|\theta_h,\theta_h}) = \sum_{i \in \mathcal{D}_\mathcal{I}} \ell(y_i, \tilde{f}(x_i, \theta_{|\theta_h}, \theta_h))\ ,\ \mathcal{D}_\mathcal{I} = \{(x_i, y_i) \in \mathcal{D},\ i \in \mathcal{I}\} \tag{II.6}$$

$\mathcal{O}_1$ typically, through successive iterations, uses $\ell_{\mathcal{D}_{train}}(\theta_{|\theta_h}, \theta_h)$ to explore the parameter space $\Theta_{|\theta_h}$ and find:

$$\widehat{\theta}_{|\theta_h} \approx \theta^*_{|\theta_h} = \arg\min_{\theta_{|\theta_h}} l_{\mathcal{D}_{train}}(\theta_{|\theta_h}, \theta_h) \tag{II.7}$$

the estimated optimal parameters given θ_h and θ'_h. Here, $\ell_{\mathcal{D}_{train}}(\theta_{|\theta_h}, \theta_h)$, when normalised by the number of examples in $\mathcal{D}_{train}$, is called the empirical risk R_{emp}.
The second optimisation algorithm $\mathcal{O}_2$ encapsulates $\mathcal{O}_1$ and uses the successive resulting $\ell_{\mathcal{D}_{train}}(\widehat{\theta}_{|\theta_h}, \theta_h)$ and $\ell_{\mathcal{D}_{val}}(\widehat{\theta}_{|\theta_h}, \theta_h)$ for finding the estimated optimal hyperparameters $\widehat{\theta_h}$ and $\widehat{\theta'_h}$ through successive application of $\mathcal{O}_1$ with different values of these hyperparameters. This meta-optimisation $\mathcal{O}_2$ is often referred to as fine-tuning.

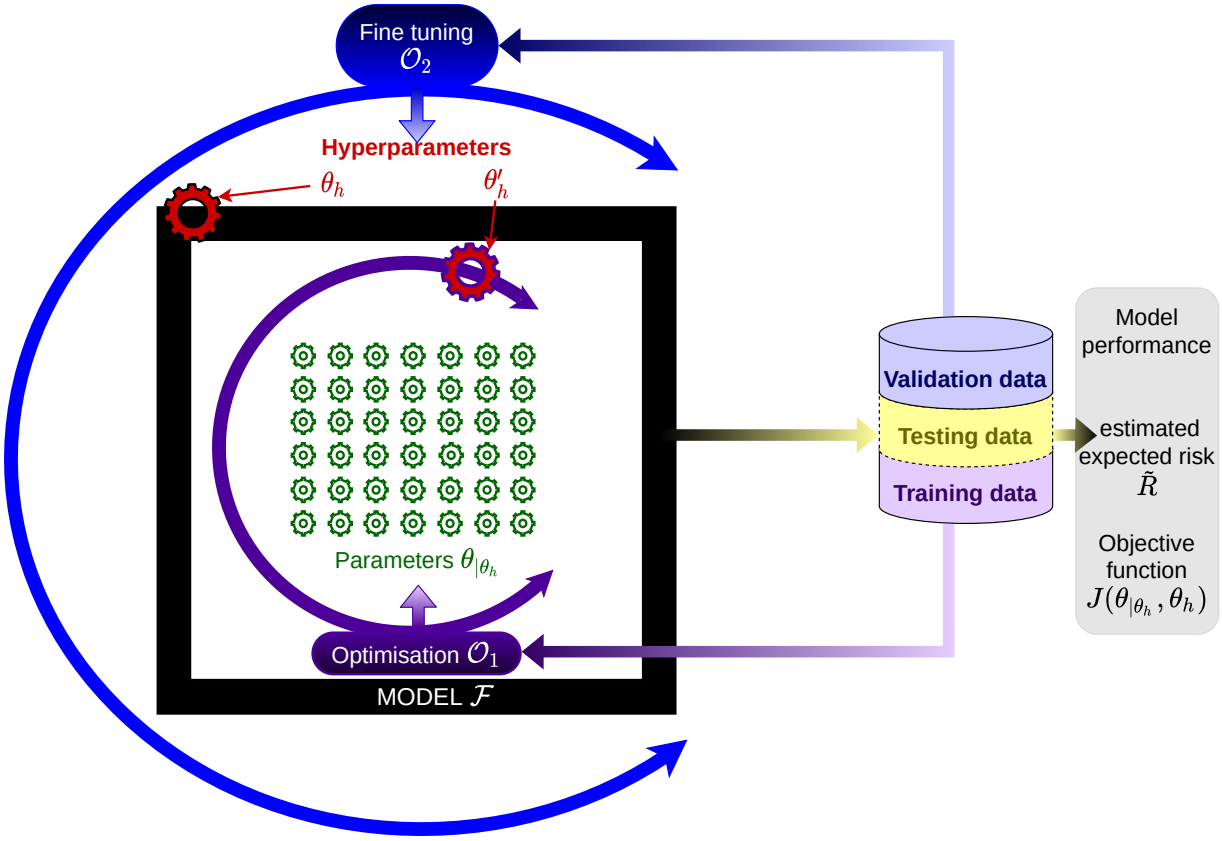


Figure II.3: Schematic of the optimisation procedure in ML

When the optimisation procedure is finished, one can estimate the expected risk R:

$$\tilde{R} = \ell_{\mathcal{D}_{test}}(\widehat{\theta}_{|\widehat{\theta_h}}, \widehat{\theta_h}) \approx \; R = \int_{\mathcal{X}\times\mathcal{Y}} \ell(y, \tilde{f}(y)p(x,y)\mathrm{d}x\mathrm{d}y \tag{II.8}$$

The end goal of optimisation is therefore to minimise R_{emp} and $\tilde{R}$. But $\tilde{R}$ being available only after the optimisation procedure — $\mathcal{D}_{test}$ being supposed to simulate unknown data, it cannot be used for optimising — the goal is therefore often reduced to minimise the loss of the validation dataset $\mathcal{D}_{val}$. In the supervised context $\tilde{R}$ is also called the objective function which is noted $J(\theta_{|\theta_h}, \theta_h)$ and it can be defined in any specific ML setting. The general optimisation procedure in ML is depicted in Figure II.3

II–2.2 Learning settings: focus on the degree of supervision

Now that the foundational notions of ML have been introduced in the previous section, focusing on a supervised setting, this section aims at describing different settings for a more complete tour of ML and for introducing important concepts used in this work. It is first important noting that machine learning problems are often categorised into two different categories, illustrated on Figure II.4, and defined hereafter:

Supervised learning problems aim at finding a relationship between data variables that are seen as inputs, i.e. explanatory variables, and data variables that are seen as outputs, i.e. explained variables or labels. The learning is thus supervised by the quality of the predictions found for outputs via the inputs [37] as presented in Subsection II–2.1.

Unsupervised learning problems, on the other hand, aim at finding relationships between data variables solely seen as inputs, these data are both explanatory and explained by themselves [55]. The data context can thus be defined as follows:

$$\mathcal{D} = \{x_i \in \mathcal{X},\ 1 \leq i \leq N\} \tag{II.9}$$

The learning is thus not supervised by the quality of the prediction for any particular variable of the data. Other criteria are therefore used for "supervising" the learning, depending on what the outputs of the model are, which are not labels. The model in unsupervised learning can very generically be defined as:

$$\mathcal{F} = \left\{ \tilde{f} : \begin{array}{ccc} \mathcal{X} & \rightarrow & \mathcal{O} \\ x & \mapsto & o = \tilde{f}(x', \theta_{|\theta_h}) \end{array} \right\} \tag{II.10}$$

where $\mathcal{O}$ denotes for the model's output space, which depends on the precise unsupervised task. In this setting, there is no generic formulation for the objective function $J(\theta_{|\theta_h}, \theta_h)$ as opposed to the supervised setting where it is defined by $\tilde{R}$. There are indeed no loss functions introduced in unsupervised ML, that compare expected output with the one actually produced by the model. Examples of specific tasks in this setting are given later in Subsection II–2.3: clustering, representation learning or dimensionality reduction.

However, these two categories are effectively two edges on a continuum of supervision. Indeed, recent framing of learning problems have emerged and can be placed somewhere in between supervised and unsupervised:

Self-supervised learning consists in framing an unsupervised learning problem such that pseudo-labels or known relationships between the data samples can be automatically extracted and used for supervision during the learning phase [61]. There are typically two types of approach in this setting, illustrated on Figure II.5 and described hereafter.
In the first type of approach, denoted here as **reconstruction learning**, the outputs are intrinsically generated from the input data X. For instance, corrupting the original data with perturbations and framing the task as reconstructing the original data examples from the perturbed one. The original data become the explained variables and the perturbed data are the explanatory variables. This setting, that is closer to supervised than unsupervised, can be formalised as follows:

$$\mathcal{F} = \left\{ \tilde{f} : \begin{array}{ccc} \mathcal{X}' & \rightarrow & \mathcal{L} \\ x' & \mapsto & l = \tilde{f}(x', \theta_{|\theta_h}, \theta_h) \end{array} ,\ \tilde{f}' : \begin{array}{ccc} \mathcal{L} & \rightarrow & \mathcal{X} \\ l & \mapsto & x = \tilde{f}'(l, \theta'_{|\theta_h}, \theta_h) \end{array} \right\} \tag{II.11}$$

where $\mathcal{X}'$, $\mathcal{X}$ and $\mathcal{L}$ respectively represent the perturbed data space, data space and latent space. Depending on the context, only $\tilde{f}$ or $\tilde{f}'$ can be of interest, or both can be. It is also possible not to necessitate a latent space $\mathcal{L}$, in which case $\tilde{f}$ and $\tilde{f}'$ are combined as a single function. For the rest, this setting is similar to the supervised one. X' corresponds to the input variables of the supervised setting and X correspond to the output ones (Y). The

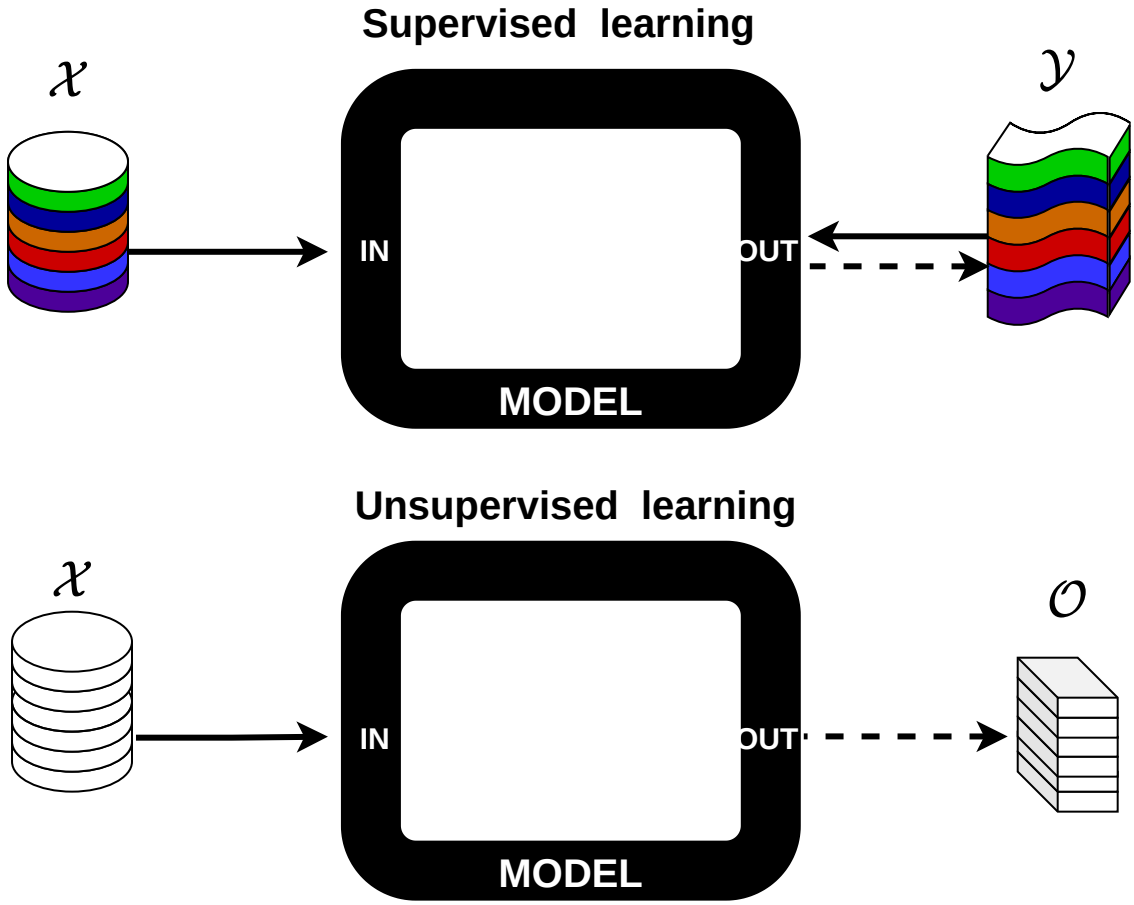


Figure II.4: Illustration of supervised and unsupervised learning setting

optimisation formulation is then similar to the supervised learning setting, but additional constraints can be added to it on the latent space.

The other type of approaches, called **contrastive learning**, uses knowledge of similarity and dissimilarity between the examples to learn a representation of these latter in a space of reduced dimension where these known relationships should be emphasised, e.g. by a distance metric that respect these similarities. To this end, pair of dissimilar examples, or triplets of two similar and one dissimilar examples, are successively compared to achieve the learning task. This setting, that is closer to unsupervised than supervised, can be formalised as follows:

$$\mathcal{F} = \left\{ \begin{array}{rcl} \tilde{f} & : \mathcal{X} & \rightarrow & \mathcal{L} \\ & x & \rightarrow & l = \tilde{f}(x, \theta_{|\theta_h}, \theta_h) \end{array} \right\} \tag{II.12}$$

$$\text{s.t. } x,\ x' \in \mathcal{X},\ r \in \mathcal{R}\ :\ d(l, l') = \tilde{r} \approx r \tag{II.13}$$

where r is a relevance score that indicate the degree of similarity between x and x', and d is a distance metric that can be fixed or parameterised and is then part of the model. $\mathcal{R}$ is the relevance score space and can be finite, e.g. $\{-1, 1\}$ with -1 and 1 respectively denoting for a dissimilar or similar pair. It can also be infinite and bounded, e.g. $[-1, 1]$ with the same signification for -1 and 1 but where r can take any value in between. More details on the contrastive learning setting are given in Chapter IV where it is used for HI construction.

Self-supervised can be useful for dimensionality reduction, representation learning, feature

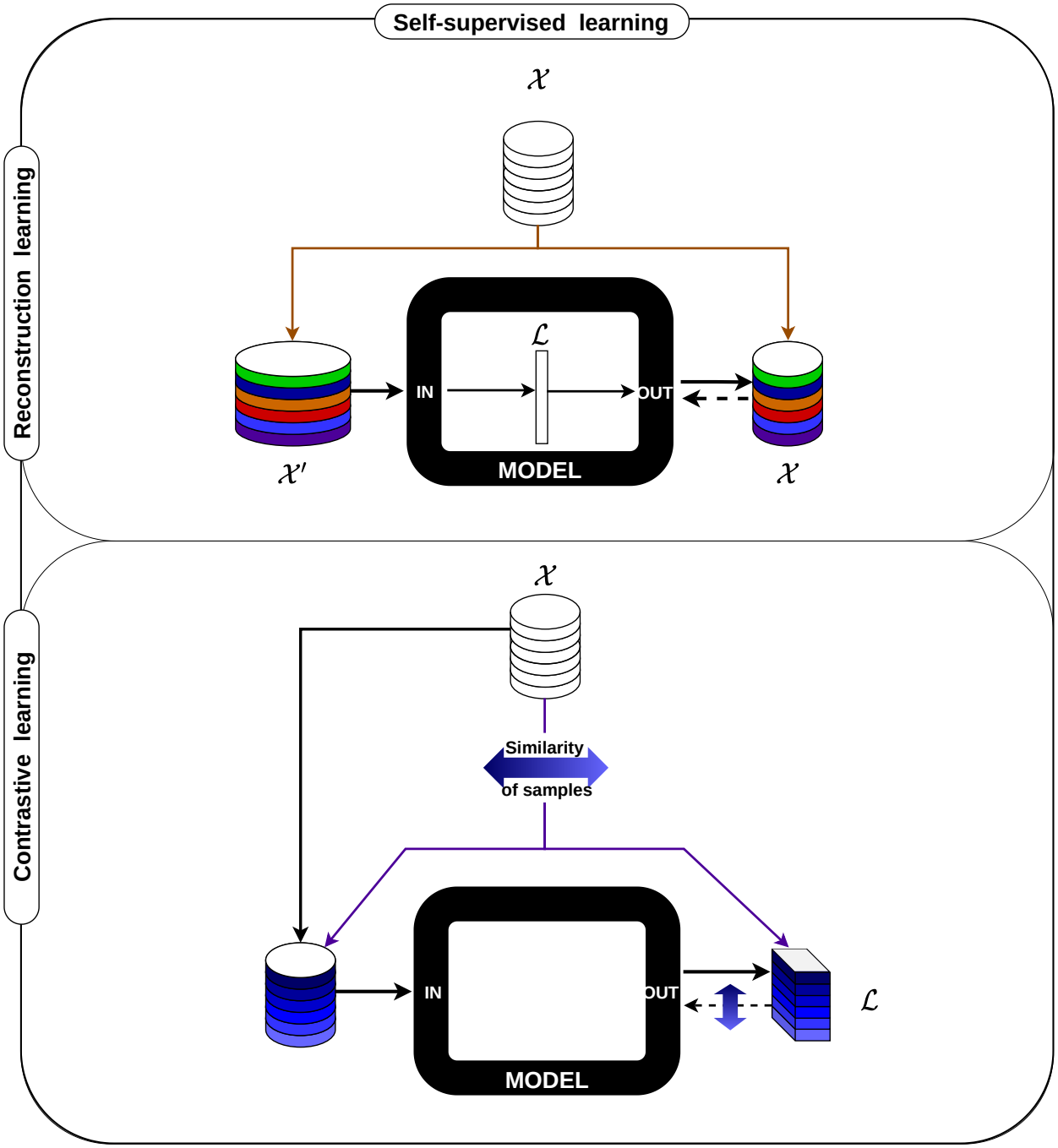


Figure II.5: Illustration of self-supervised learning setting

extraction, or data denoising. These self-supervised learning problems are often used as **pretext** tasks that allow a model to extract general and relevant features from the data, features that can be exploited to solve downstream tasks, most often in a supervised setting [61].

Semi-supervised learning is a similar setting as supervised learning, directly influenced by a context of application in which a large amount of data is unlabelled. This means that the goal of the learning task is still to map inputs to outputs as in supervised learning, but output

data, i.e. the labels $y_i \in \mathcal{Y}$ are not available for each input data $x_i \in \mathcal{X}$:

$$\mathcal{D} = \left\{ \quad (x_i, y_i) \in \mathcal{X} \times \mathcal{Y},\ 1 \leq i \leq M \quad (x_j) \in \mathcal{X},\ 1 \leq j \leq N \geq M \right\} \tag{II.14}$$

Usually, only a few samples of input data, M, are given with their own labels. Semi-supervised undertakes to first fit a model on available pairs of inputs and outputs. Then the model is used to create pseudo-labels on the remaining inputs deprived of true labels. Usually, the process of pseudo-labelling is iterative, pseudo-labels are associated with a confidence level and only pseudo-labels with high confidence are used as labels for the next iteration, the model is refined with the originally available pairs of samples and labels, as well as the newly generated pairs of samples and pseudo-labels. This process, illustrated in Figure II.6, is repeated until the confidence of pseudo-labels converges [194]. This setting only differs from the supervised one in its particular iterative optimisation process, where more and more unlabelled data examples are assigned a pseudo-label at each iteration. The only difference in the model definitions is that semi-supervised learning necessitates a confidence score for each predicted label in order to decide which pairs of examples and pseudo-labels can be added to the training data for the next iteration:

$$\mathcal{F} = \left\{ \tilde{f} : \begin{array}{ccc} \mathcal{X} & \to & \mathcal{Y} \times \mathcal{C} \\ x & \mapsto & (y, c) \end{array} \right\} \tag{II.15}$$

Where $\mathcal{C}$ denotes for the confidence score space. This one can be e.g. $\mathcal{C} = [0, 1]$ for a discrete $\mathcal{Y}$ space (classification) where each c represents the confidence as a probability, or $\mathcal{C} = \mathbb{R}$ for a continuous $\mathcal{Y}$ space (regression) where each c represents the variance of a normal distribution around the predicted value

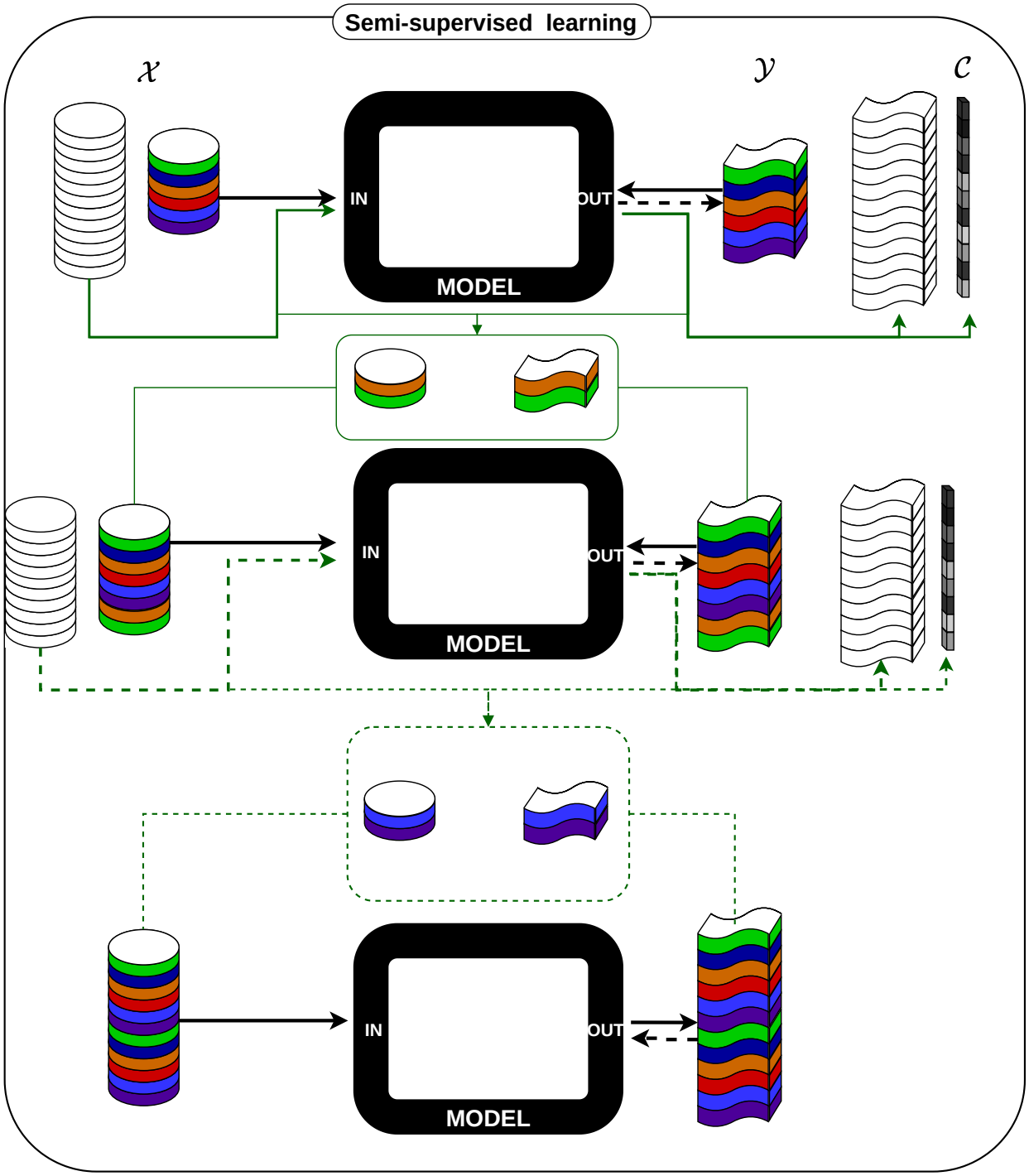
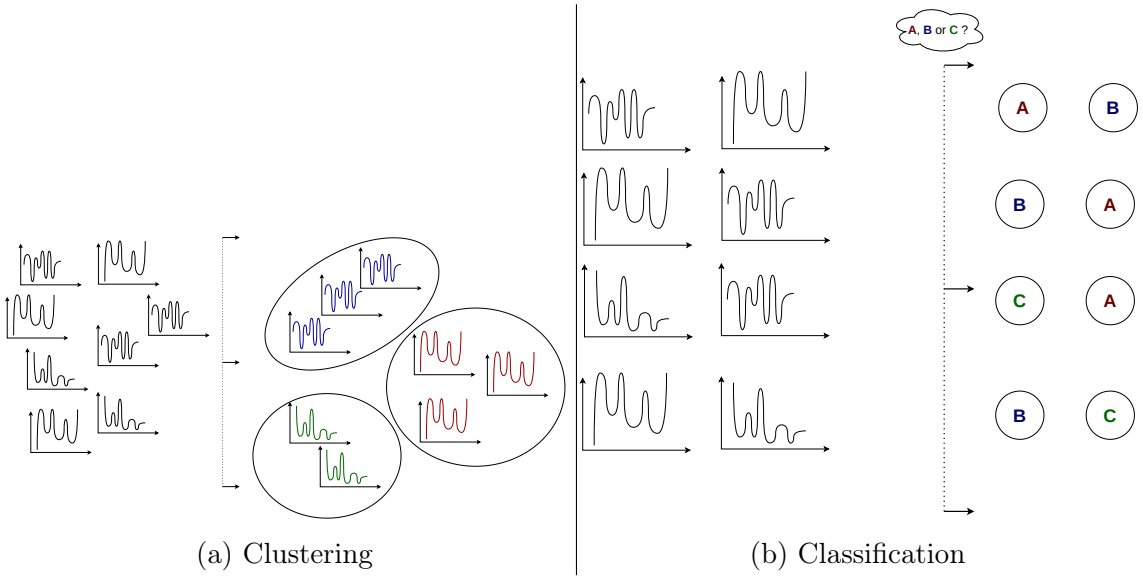


Figure II.6: Illustration of semi-supervised learning setting

II–2.3 Downstream learning tasks on MTS

This section discusses the fundamental learning problems on MTS: clustering, classification, representation, and regression. Each of these problems serves a distinct purpose and offers insights into different aspects of the data. The description of the main types of learning tasks is thus carried

on, with a particular attention on where they belong to on the supervision spectrum. Additionally, the context of MTS data is now introduced, the data space of model inputs $\mathcal{X}$ in particular is now denoted $\mathcal{S}$ as the space of MTS or MTS subseries of d variables and of length l.



(a) Clustering (b) Classification

Figure II.7: Illustration of the clustering and classification tasks

Clustering

Clustering involves grouping similar MTS instances $s^{(k)} \in \mathcal{S}$ or segments $s^{(k)(i)} \in \mathcal{S}$ together based on their inherent characteristics or patterns, as shown on Figure II.7a. It belongs to the far edge of unsupervised learning in the supervision spectrum. It helps identify hidden structures within the data by discovering clusters or subgroups that exhibit similar behaviours across the variables. It therefore requires a discrete univariate output indicating the cluster κ to which a data sample belongs [162], $\kappa \in \mathcal{O} = \{1, \dots, K_{\tilde{f}}\}$ where $K_{\tilde{f}}$ the number of clusters depending on the model $\tilde{f}(\cdot, \theta_{|\theta_h}, \theta_h)$. This learning setting can be expressed as follows:

$$\mathcal{D} = \{s^{(k)} \in \mathcal{S}\}_{k \in \mathcal{K}} \text{ or } \{s^{(k)(i)} \in \mathcal{S}\}_{k \in \mathcal{K}; i \in \mathcal{I}_k(T^{(k)})} \tag{II.16}$$

$$\mathcal{F} = \left\{ \tilde{f} : \begin{array}{ccc} \mathcal{S} & \rightarrow & \mathcal{O} = \{1, \dots, K_{\tilde{f}}\} \\ s & \rightarrow & \kappa = \tilde{f}(s, \theta_{|\theta_h}, \theta_h) \end{array} \right\} \tag{II.17}$$

Clustering can be particularly useful for exploratory data analysis, anomaly detection, and identifying patterns in MTS data [178], thus providing a comprehensive understanding of the system under study and uncovering meaningful relationships and dependencies that may not be evident through an individual variable analysis alone.

Classification

Classification, by contrast, focuses on assigning MTS instances to predefined classes or categories, also called labels, based on their characteristics, as shown on Figure II.7b. It is, in that sense, the supervised version of clustering. Let us denote, $\mathcal{Y} = \{1, \dots, K\}$ where K the

number of classes. This setting, that can be expressed as follows:

$$\mathcal{D} = \{(s^{(k)}, y^{(k)}) \in \mathcal{S} \times \mathcal{Y}\}_{k \in \mathcal{K}} \tag{II.18}$$

or (II.19)

$$\mathcal{D} = \{(s^{(k)(i)}, y^{(k)}) \in \mathcal{S} \times \mathcal{Y}\}_{k \in \mathcal{K};\, i \in \mathcal{I}_k(T^{(k)})} \tag{II.20}$$

$$\mathcal{F} = \left\{ \begin{array}{rccll} \tilde{f}: & \mathcal{S} & \rightarrow & \mathcal{Y} & \\ & s & \rightarrow & y & = \tilde{f}(s, \theta_{|\theta_h}, \theta_h) \end{array} \right\} \tag{II.21}$$

is particularly useful when the objective is to classify new instances into known classes or predict their behaviour based on historical data, in particular when this behaviour is highly dependent on the class of the instance under study. By tackling classification tasks, one can make predictions about the nature of an instance of the system under study and thus more adequate decisions based on the characteristics and patterns observed in MTS data. This enables the identification of specific states or events based on the behaviour of the variables, which can be valuable in various domains, such as engineering [229, 63] or healthcare [132].

Representation

Representation learning aims at projecting input data into a different space of higher or lower dimension. It can be thought of as an unsupervised learning problem, but is often set up as a self-supervised learning one. In contrast to clustering and classification, the output space, also called latent space $\mathcal{L}$ of a representation learning task is expected to be continuous and not discrete. This setting is particularly adequate for dimensionality reduction, where the purpose is to combine the existing dimension into a lower number of dimension while still explaining most of the variance between the data samples or emphasising known relationships and similarities. In a context where the data consist of MTS, the data samples have dimensions equal to the number of variables d times the number of time steps l, these dimensions are often cumbersome to process and redundant. Representation learning tasks, such as dimensionality reduction, are therefore of particular interest, even as a preprocessing or pretext task [50], which is the reason the output space is called latent space. The representation learning setting, illustrated on Figure II.8, can be formulated as follows:

$$\mathcal{D} = \{s^{(k)} \in \mathcal{S}\}_{k \in \mathcal{K}} \text{ or } \{s^{(k)(i)} \in \mathcal{S}\}_{k \in \mathcal{K};\, i \in \mathcal{I}_k(T^{(k)})} \tag{II.22}$$

$$\mathcal{F} = \left\{ \begin{array}{rccll} \tilde{f}: & \mathcal{S} & \rightarrow & \mathcal{L} & ;\, \mathcal{L} = \mathbb{R}^L \\ & s & \rightarrow & \ell & = \tilde{f}(s, \theta_{|\theta_h}, \theta_h) \end{array} \right\} \tag{II.23}$$

where L is the dimension of the latent space $\mathcal{L}$. Representation learning can be thought of as the non-supervised version of multivariate regression, which is presented hereafter.

Regression

Regression involves predicting or modelling the relationship between variables in an MTS. Regression tasks are framed in a supervised or semi-supervised setting. In the context of MTS analysis, regression can be further categorised into two distinct types: extrinsic regression and intrinsic regression.

Extrinsic Regression refers to modelling the relationship between a MTS that is considered to be composed of explanatory variables $\mathcal{X} = \mathcal{S}$ and different signals or variables that are presumed to be explainable by the former $\mathcal{Y} = \mathcal{S}'$ as illustrated in Figure II.9a.

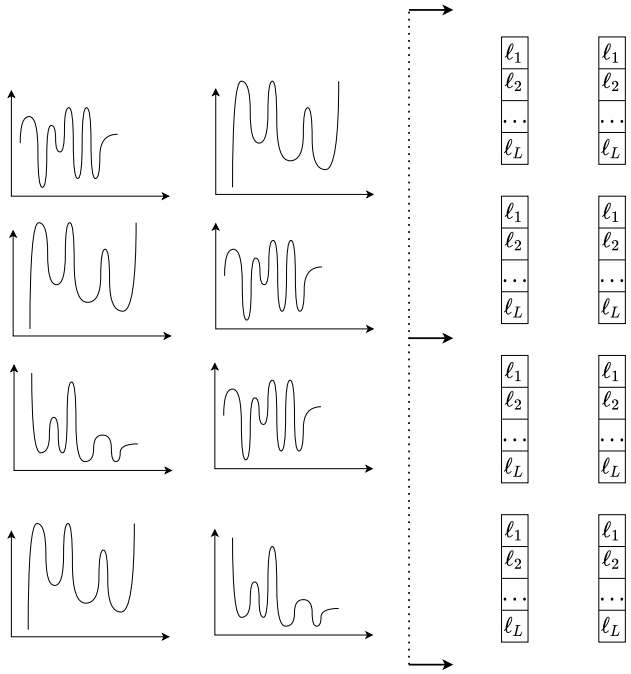


Figure II.8: Illustration of representation learning

By establishing a relationship between the MTS and the extrinsic variables, valuable insights can be gained regarding the system dynamics and its response to external factors or other internal conditions. The extrinsic regression of time series can be formulated as follows:

$$\mathcal{D} = \{(s^{(k)}, s'^{(k)}) \in \mathcal{S} \times \mathcal{S}'\}_{k \in \mathcal{K}} \text{ or } \{(s^{(k)(i)}, s'^{(k)(i)}) \in \mathcal{S} \times \mathcal{S}'\}_{k \in \mathcal{K};\, i \in \mathcal{I}_k(T^{(k)})} \tag{II.24}$$

$$\mathcal{F} = \left\{ \tilde{f} : \begin{array}{ccccc} \mathcal{S} & \rightarrow & \mathcal{S}' & & \\ s & \rightarrow & s' & = \tilde{f}(s, \theta_{|\theta_h}, \theta_h) \end{array} \right\} \tag{II.25}$$

where s and s' are MTS issued from two different stochastic processes. A common setting of extrinsic regression is to defined one or several QoIs of the system under study as the variables to be explained, and other internal signals of the system as well as additional external signals as the explanatory variables. Extrinsic regression models are often employed in areas such as environmental monitoring, where the MTS variables are influenced by external factors like weather conditions, pollution levels, or socio-economic indicators [186].

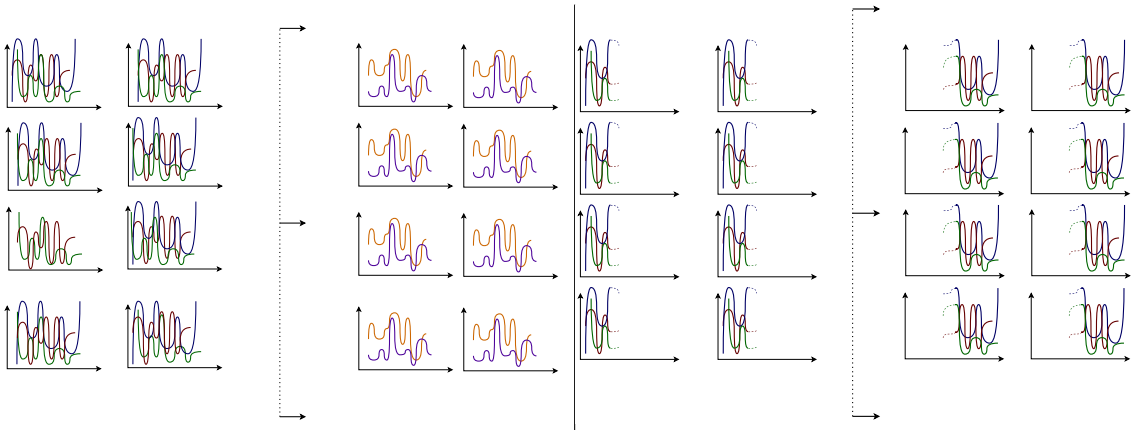
Intrinsic Regression focuses on addressing specific tasks within the MTS domain, such as missing values imputation, forecasting and backcasting, denoising, smoothing or trend extraction. These tasks involve predicting or estimating the values of variables within the MTS based on historical observations of the same MTS potentially corrupted by noise. Intrinsic regression models take into account the temporal dependencies and relationships between the variables to make accurate predictions. An illustration of intrinsic regression, and more particularly forecasting, is given on Figure II.9a. This

setting can be expressed, in the context of forecasting as follows:

$$\mathcal{D} = \{(s^{(k)}, s'^{(k)}) \in \mathcal{S}_l \times \mathcal{S}_{l'}\}_{k \in \mathcal{K}} \tag{II.26}$$

$$\mathcal{F} = \left\{ \begin{array}{rccl} \tilde{f}: & \mathcal{S}_l & \rightarrow & \mathcal{S}_{l'} \\ & s & \rightarrow & s' \quad = \tilde{f}(s, \theta_{|\theta_h}, \theta_h) \end{array} \right\} \tag{II.27}$$

where s and s' are time series issued from the same realisations of a common stochastic processes, and where, in the context of forecasting, $\mathcal{S}_l$ denotes for MTS of length l and $\mathcal{S}_{l'}$ for MTS of length l' which are the future of $\mathcal{S}_l$. Missing values imputation aims to fill in the gaps or missing values within an MTS, enabling to maintain the continuity and integrity of the data. Forecasting involves predicting future values of the MTS variables, allowing for proactive decision-making and trend analysis. Backcasting, on the other hand, entails estimating past values of the MTS variables, which can be valuable for historical analysis or data reconstruction. Denoising of smoothing denotes for the identification and adjustment of potentially erroneous or noisy data, while trend extraction refers to the retrieval of general trends in a MTS ignoring small fluctuations and seasonality that are less significant. By utilising intrinsic regression techniques, one can gain a deeper understanding of the temporal behaviour of the MTS variables and make informed predictions or estimations.



(a) Extrinsic regression (b) Intrinsic regression (here forecasting)

Figure II.9: Illustration of the MTS extrinsic and intrinsic regression tasks

II–2.4 Learning models and estimation of their parameters

As presented just above, a learning model $\tilde{f}$ is essentially a computational model, i.e. a functional of inputs x, which is parameterised by parameters θ_{θ_h}, and produces outputs y. This functional can be linear or non-linear on x and deterministic or stochastic. The nature of the model, and its architecture, are partially determined by the problem to be solved. If one wishes to predict the class of any data record in a dataset, its model needs to be designed in such a way that it outputs some representation of a class, i.e. one-hot representation. And the same holds for any other problem, it determines the model to output a specific mathematical object. Similarly, the nature and characteristic of the data to be processed also partially determines the architecture of the model.

Nevertheless, there remains a large a degree of freedom in how to design the model, in terms of architecture and hyperparameters. Some are tailored for specific learning problems and others are so general that they can adapt to any of those.
This section first aims at giving a brief overview of some generic existing modelling methods with a focus on MTS learning in a first part. Mathematical definitions and details are omitted since such methods have not been particularly reused in the present work, except the state space model (SSM). The bibliography on the specific usage of such learning models for PHM related tasks is outlined later in Chapter III. In a second part of this section, a brief overview of generic optimisation or parameter estimation is given, similarly as for the first part of this section, the mathematical definitions and details are omitted. Neural network models are purportedly kept out of this section, and dedicated to the entire next section, as they will be the basis of future developments.

II–2.4.1 Model's overview

Linear deterministic models, such as linear regression the simplest form of, SVM (Support Vector Machines) [183], PCA (Principal Component Analysis) [18, 212], offer simple, lightweight, computationally efficient and interpretable models. However, they lack the possibility to fit complex non-linear problems, such as the one often encountered in MTS learning. Linear models, specially crafted for time series, do exist however, they mainly aim at applying linear interaction with respect to time, i.e. dependency between current observation and the ones at previous time lags. These models, however, differ from the classical linear models presented here for their stochastic nature and distinct modelling approaches, which are more related to stochastic process theory, they are therefore introduced later in **autoregressive-models**.

Kernel methods, introduce non-linearity to linear models by mapping original data into a transformed and higher-dimensional feature space [183], a solution known as the "kernel trick". A kernel function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ applied in the original space $\mathcal{X}$, is in fact equivalent to an inner product in a higher dimensional space. These kernel functions can therefore be interpreted as similarity or distance measures [169] in this higher dimensional space. They enable linear models to capture intricate nonlinear dependencies in the data, more precisely, they allow linear models to solve the problem at hand in a higher-dimensional space, where linear separation or regression is possible. While kernel methods have efficiently adapted linear techniques for non-linear problems, their application for MTS has not been very practical so far, but quite fruitful for UTS [3, 221]. However, some existing works have investigated this modelling approach for MTS and discovered specific kernel models that perform better than classical kernels in MTS [38, 187]. Nevertheless, these investigations remain a minority in MTS modelling for machine learning, and kernel-SVM are thus more used in UTS learning tasks.

Decision trees, represent a very different modelling approach that employs the hierarchical structure of a tree, or directed acyclic graph, that recursively separates input samples based on their data features, providing interpretable decision-making flowcharts [90]. Decision trees are interpretable and can handle both categorical and numerical data, making them very handy for situations of poorly structured data. This modelling technique has the particularity of not requiring a fixed architecture with parameters to be optimised, the precise architecture of

the tree is built during the optimisation process. If classical decision tree can solve some UTS learning tasks like forecasting they are not of particular interest for learning on MTS, but recent developments attempt to adapt decision trees to such data [143, 171]. Additionally, decision trees are often used as base learners in an encapsulating modelling approach, proved to be efficient in time series forecasting, described hereafter.

Ensembling, is a modelling technique that seek to enhance existing methods, just like kernel methods. It combines multiple individual models to improve the overall predictive performance [165], hence the individual models do not necessarily need to be experts model but can be basic weak learners. By leveraging the wisdom of multiple models, ensembling aims to achieve higher accuracy and robustness. Ensemble modelling is quite straightforward, the only model hyperparameters are the size of the ensemble, the nature of the individual models and the ones defining the aggregation method of the ensemble. The main differences between ensembling methods therefore rely more on their specific optimisation procedure than their modelling. Because they can be weak learners or expert models depending on their sizes, decision trees are the most used individual models for ensembling [165]. They give the most famous and performant ensemble techniques: random forest [15], Adaboost [168] and gradient boosting trees [30, 84]. The latter one, proved to be very efficient for processing time series data, especially in forecasting tasks, as shown in the M5 forecasting competition where gradient boosted trees were used by most of the highest ranking models [74]. The recent efforts to adapt basic decision trees to temporal data and the high performance of ensembling techniques using decision trees, indicates a promising potential of these methods to learn on MTS.

Gaussian processes, are stochastic processes whose collection of random variables are such that any subset of them has a multivariate normal distribution [173]. An interesting property of Gaussian processes is that they are fully defined by their means and covariance functions. Particularly, the latter are at the core of Gaussian process modelling and can be related to kernel methods. If Gaussian processes' foundational theory relies on stochastic processes and therefore is of particular interest for studying time series data [161], it is also largely used and developed in a more general continuous regression framework regardless of its indexation on time, in that sense it is often referred to as kriging [36]. For this work, however, their rooting in stochastic process modelling is what make Gaussian processes a referential modelling practice for learning on time series data [52].

Bayesian networks , is another important modelling method used in machine learning. They represent probabilistic relationships between variables through directed acyclic graphs. More precisely, they represent the conditional dependencies between variables. Each node of the graph represents a variable, be it an observed data variable, a latent variable, a noise, or other hypotheses variable [66]. Each branch going from one node to another represents the conditional dependency of the target node on the parent one. It is a very generic method for modelling in machine learning, the ones encompassed in one specific variant: dynamic Bayesian networks (DBN), are especially utilised in stochastic process modelling, which are of particular interest for this work. Various attempts have been proposed to model different types of stochastic processes as dynamic Bayesian network (DBN) [142, 89, 133]. If DBNs can be thought of as a generalisation of more particular concepts it can represent, in particular

hidden Markov models or Kalman filter models [133], these particular concepts have been mostly developed before or aside of the Bayesian networks theory and have their own theories and practices. These latter include the remaining types of model described hereafter.

Autoregressive (AR) models, aim at representing a stochastic process realisation as a function of its time-lagged values and time-lagged values of a white noise [137]. AR moving average (ARMA) models are the linear case of such models and mostly limited to model correctly stationary univariate stochastic processes [137]. Extended formulation of ARMA attempt at bypassing these limitations with varying degrees of success. Vector ARMA (VARMA) allows the representation of multivariate stochastic processes by representing a realisation of a stochastic process not as an ordered collection of values, but as vectors where each dimension is mutually interacting [116]. Autoregressive Integrated Moving Average (ARIMA) models [137] enable to take into account non-stationarity in terms of the process' mean, i.e. only a trend of the mean can be represented. Similarly, seasonal ARIMA (SARIMA) can model a seasonality in the mean evolution [94]. In that sense, ARIMA and SARIMA do succeed in partially solving the limitation of stationarity with classical ARMA. Nevertheless, ARMA models and their extensions can not model non-stationarity of higher moment statistics of a stochastic process like the variance, for instance. It still has also the limitation caused by its simple modelling approach, that is, it can only depict linear relationships between time-lagged values. Moreover, ARIMA models have been mainly developed for modelling a stochastic process of a single observation of time series and therefore lack methods for describing a dynamic system if multiple realisations of a similar stochastic process are available. Non-linear versions of AR models also exist, Nonlinear autoregressive with exogenous input (NARX) is a generic formulation of such models where the dependances between time-lagged values can be non-linear, moreover this modeling approach is particularly adapted to MTS because it accounts for exogenous (or extrinsinc) inputs, that is different variables of a MTS [120].

State-space models (SSM), At their core, describe the evolution of latent, unobservable states usually through discrete time steps, coupled with observed measurements that are influenced by these states [6]. A SSM consists of two key components: the state equation and the observation equation. The state equation defines the evolution of latent states according to a stochastic process. The observation equation links the observed measurements to the underlying states through a linear or nonlinear relationship, possibly subject to measurement noise. The combination of these equations constitutes a dynamic system that can capture a wide range of temporal behaviours. If the future state of the process depends only on its current state and not on the sequence of states that preceded it, then the SSM is referred to as a hidden Markov model (HMM), i.e. its states' equation satisfies the Markov property [118]. In that case, the state equation represents a Markov chain. A Markov chain can be constructed as a Bayesian network but can not be thought of as a state-space model as it does not necessitate the construction of a latent variable. Markov chain models are often used to model situations where the system's dynamics are fully observable and can be directly represented by the sequence of states, whereas HMMs find applications in scenarios where the underlying states are unobservable, and the observations are noisy or incomplete. The relationship depicted in the observation and state equations of a SSM can be linear and Gaussian (i.e. noises are considered as white noise) in which case the model is referred to

as linear Gaussian SSM or Kalman filter models [207]. More complex SSM can also be built and optimised on with non-linear state and observation equations, which are more adapted to complex cases of MTS modelling [39, 156].

Lévy processes, refer to a class of stochastic processes with independent and stationary increments [10]. They are non-stationary by construction, due to their random increments. These processes are therefore additive and do possess the Markovian property. They can therefore be modelled as DBN, but are clearly not studied in this framework. They are particularly useful when studying complex and chaotic dynamical systems . Machine learning models based on Lévy process have found application in finance or environmental sciences [10] and, more interestingly for this work, in predictive maintenance to model machinery degradation processes where they are often refered to Wiener process and combined with non-linear drift functions [101, 175, 188, 189].

II–2.4.2 Optimisation or parameters estimation for learning models

The models themselves however are not performing well by nature, they needs to be optimised, and to have their parameters appropriately estimated to solve at best the problem at hands. An important common ground of parameter estimation processes is that they somehow need a stopping criterion. If for some specific cases an optimal solution can be found and recognised, generally there is no evidence for such an achievement. These optimisation techniques rely then on performance indicators, often called **loss function** and presented in Subsection II–2.1. These loss functions need to be minimised, and stopping criteria vary from a threshold value above which we consider a solution satisfying to a threshold on the rate of improvement of successive solutions, i.e. convergence, or the absence of loss improvement of new solutions.
For each model, several optimisation methods exist, but any optimisation method cannot be used for any model. Sometimes a model and its parameter estimation procedure are so tightly related that separating the two is not really valuable. This separation is here made in a conceptual attempt at encompassing various existing learning approach in a relatable scheme.
For linear models the optimisation of parameters can always be formulated as a convex, linear or quadratic, optimisation [32], and special methods are available for this kind of optimisation, such as interior point method [136]. These models however can also be optimised using other optimisation algorithm like gradient descent or least-squares based methods.
Least-squares based methods directly aim at minimising the sum of squares of the errors (or residuals) and they take the form of iterative algorithms that use derivatives and often second derivatives of the model with respect to its parameters to improve the model. It can not guarantee convergence, but if it converges, in the best case, it will be toward the optimal solution, or at least to a local optimum [80]. The most used least-squares based methods are the Gauss-Newton [203] and Levenberg-Marquart [129] algorithms.
Gradient descent is a more general optimisation method in the sense that it aims at minimising any cost function, or loss function, and not just the squared residuals. [164]. It is also an iterative method, but that only use the gradient (first derivative) of the loss function as a guide for exploring the space of solutions. At each step, the gradient of the loss function for each parameter at its current value indicates the direction where better solutions can be found.
Each of these methods implies searching for optimal parameters in a explicit and rational fashion that can ensure at least a local optimum. These methods can be improved by so-called **regulari-**

sation techniques [179]. These latter often impose some conditions on the solution of parameters, like upper and lower bounds, norms, or sparseness. They can be interpreted as hypotheses on the form of the solution, or as a way to facilitate the finding of an optimal solution. In that sense they can also be regarded as heuristic, i.e. a method that can find approximate solution of problems for which well-defined classical methods do not exist or are computationally prohibitive. [58]. The ideal or true solution does not necessarily follow from this heuristic or can even potentially be disfavoured or avoided, but in most cases it enables one to find acceptable solutions in a reduced amount of time.
If heuristics are just a way of improvement for the optimisation methods mentioned above, some parameters estimation approaches rely solely on heuristics and are therefore called heuristic optimisation. In general, however, relying on heuristics alone is not advisable, as heuristics often have no convergence properties and are extremely context-dependent. [177].
For optimising models when none of the above techniques are adequate, i.e. the problem cannot be solved entirely in a reasonable amount of time neither by linear nor quadratic programming, least-squared methods, gradient methods nor available heuristic, one might still resort to meta-heuristic methods. Meta-heuristic search methods are upper level general methodologies that can be used as guiding strategies in designing underlying heuristics to solve optimisation problems [42]. The most famous meta-heuristic methods are inspired from natural phenomena that they try to mimic: i.e. ant colony [44], particle swarm [85], genetic algorithm [127] or simulated annealing [46]. These approaches often rely on iterative procedures of probabilistic exploration of the solution space, simulation of potential candidate solutions and assessment of performance of these solutions. A very common type of meta-heuristic optimisation is, in that sense, the Monte-Carlo based parameter estimation [115].

II–2.5 A focus on neural networks

In this section a focused overview on neural networks (NN) models is given, as it is the type of model explored for solving the PHM tasks in this thesis.

II–2.5.1 Overall presentation and use for time series

Neural networks (NN), is another modelling method, inspired by the structure of animal's brain. It consists of interconnected layers of artificial neurons. Each neuron computes a linear mapping between its parameters and its inputs and apply a non-linear activation function to this mapping. Each layer's output is the input of the next one. The resulting model yields complex non-linear interaction between variables of the input data which renders it really opaque to human understanding, i.e. it is generally not interpretable [56] even though recent development undertakes to tackle this issue. This very generic modelling approach can capture complex nonlinear relationships among input variables and, or with, output predictions, which makes them very efficient in practice. It encompasses some specific NN architecture which distinguish themselves by the nature of the layers they use. In particular, NN have gained significant attention due to their ability to handle large-scale datasets and solve intricate problems like image [91] and speech recognition [41] or natural language processing (NLP) [196]. This last domain of application raised significant attention and development to specific neural network architectures. First, recurrent neural networks (RNN) have

been developed and specialised in the processing of sequential data, which makes them of particular interest for UTS and MTS data. A new NN-based NLP paradigm has recently emerged, known as the attention mechanism [195], leading to transformers, which are present in most high-performing large-language models (LLM). Interest in the use of attention mechanisms for time series data learning has therefore been raised [109, 209, 222] due to their performance in the sequential data contexts of NLP. To learn about MTS, investigating NN and their sequence-specific modes: RNN and attention mechanisms, is of particular interest.

II–2.5.2 Theoretical considerations

(a) Modelling

As mentionned above, a NN is a combination of subsequent layers of neurons. A layer, possesses multiples neurons, each linearly combining the inputs of the layer with its own parameters to obtain a single value that is then passed through an activation function a to produce the output of the neuron. A single neuron, also called perceptron or unit, here denoted u is defined as follow:

$$y = u(x) \tag{II.28}$$

$$= a(\sum_{i=1}^{n} w_i x_i + b) \tag{II.29}$$

With y and x respectively the output and inputs of the neuron, w and b respectively the weights and bias of the neuron, i.e. its parameters.
The most simple type of NN layer is a dense layer, it is composed of n_u neurons also called units. That each takes as input the layer's input. The dense layer l is therefore defined as follow:

$$y = l(x) \tag{II.30}$$

$$= [p_j(x)]_{j=1}^{n_u} \tag{II.31}$$

With y and x respectively the outputs and inputs of the layer, and u_j a neuron. Finally, the most simple form of NN architecture is the multi-layer perceptron (MLP) which is composed of N_l subsequently connected dense layers, i.e. the first layer outputs are the second layer inputs, and similarly until the last layer. A MLP is then defined as follow:

$$y = MLP(x) \tag{II.32}$$

$$= l^{(N_l)} \circ l^{(N_l-1)} \circ \cdots \circ l^{(k)} \circ \cdots \circ l^{(1)}(x) \tag{II.33}$$

With y and x respectively the outputs and inputs of the MLP, and $l^{(k)}$ a layer of neurons. An example of a MLP architecture is given in Figure II.10.

More complex architectures of MLP exist, and are also used in this work, this complexity can arise by connecting layers to multiple subsequent ones, or by using different types of layers. Can e.g. be mentioned recurrent and convolutional layers.

Recurrent layers consist of layers that treat inputs that are sequential, i.e. the input has a dimension of variables and a time dimension, and the variables are fed to the layer, time step per time step. The recurrent layer has the particularity of using its previous time-step outputs as inputs

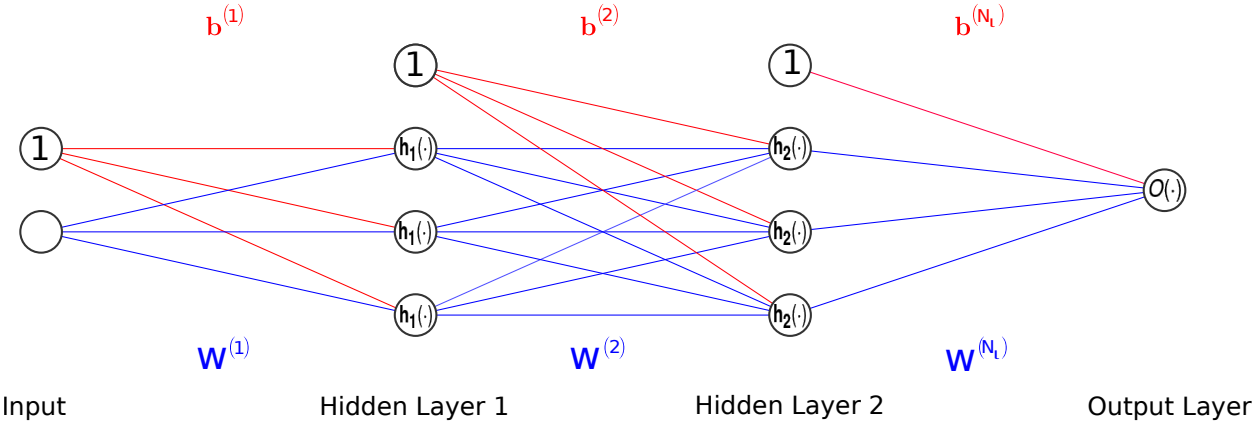


Figure II.10: Example of MLP architecture with 2 layers, 3 neurons per hidden layer, one input and one output.

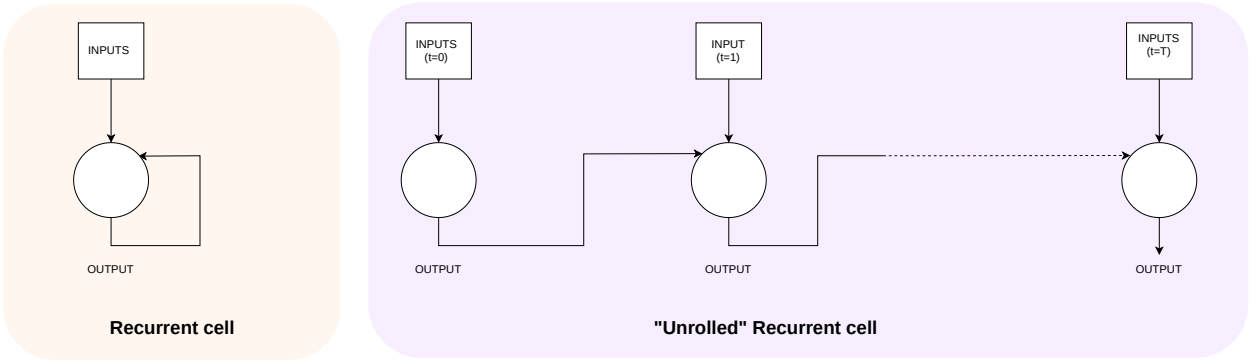


Figure II.11: Illustration of a recurrent neuron.

to the current time step. Many types of recurrent layers that produce different types of outputs, e.g. states or memories, exist. They go beyond the classical implementation, long-short term memory (LSTM) and gated recurrent unit (GRU) can e.g. be cited. The aim of this section however is not to describe in details these latter. A visual representation of a recurrent neurons can nevertheless be found in Figure II.11

Convolutional layers consist of layers whose neurons are convoluted through the inputs instead of simply mapping all the inputs to the neurons parameters. This reduces drastically the number of parameters per layer, and can be very efficient for large and multi-dimensional inputs like images. Once again, this type of layer is not more thoroughly described as it is not the aim of this section, but a visual representation of a convolutional neuron can be found in Figure II.12

Some other types of layer, which have no parameters, also exist, they only aim at facilitating the learning procedure, improving the generalisation ability of the NN, or facilitating transition between different types of parameterised layers. Can be cited, e.g. dropout layers, pooling layers or flattening layers. Dropout aims at randomly filtering some of its input before being propagated to the next layer, and is helpful for improving generalisation and can hence be seen as a regularisation technique. Pooling layers aim at reducing the size of inputs / outputs transiting in-between each parameterised layer by aggregating the ones that are adjacent, via average, maximum or minimum operations, these layers hence reduce the number of parameters of the NN, and are also improving

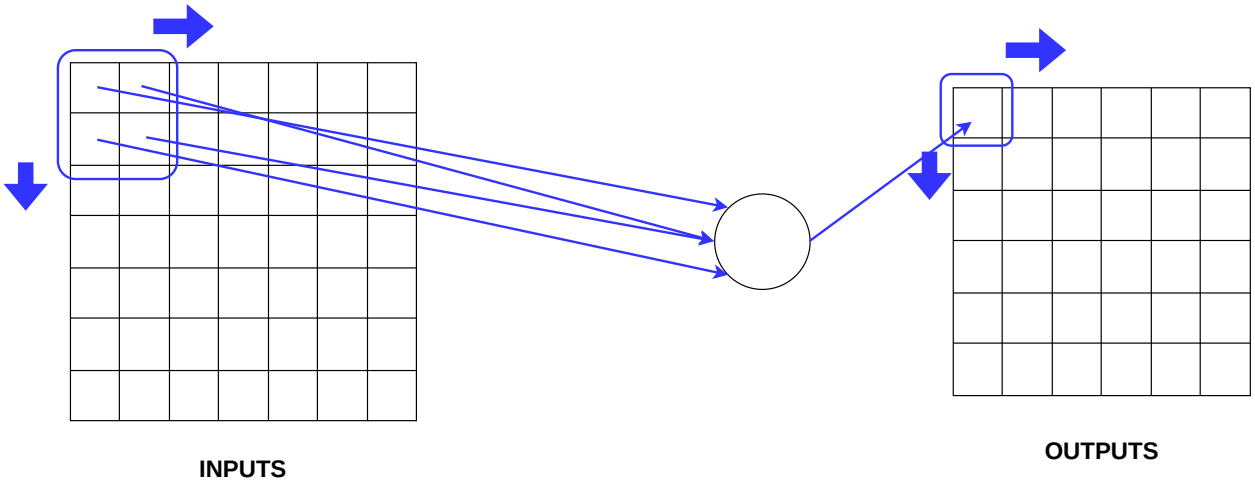


Figure II.12: Illustration of a convolutional neuron.

the generalisation ability especially for convolutional neural networks (CNN). Flattening layers aims at transforming layer's output that are not in the form of vectors, into vectors. Convolutional and recurrent layers typically have non-vector outputs, respectively multi-dimensional tensors (2 or 3 dimensions) and time-distributed vectors (equivalent of matrices). These flattening layers are thus necessary for transitioning from recurrent or convolutional layers to dense ones.

The main idea to bear in mind, is the inter-connection of more or less wide layers in more or less deep neural network architecture. The precise architecture of a NN is always challenging to optimise, some experimental and theoretical results exist privileging the choice for certain types of layers and, to some extent, their inter-connection, depending on the situations and data contexts. Nevertheless, the precise development of an architecture is also often based on trials and errors, and on hyperparameters tuning. These modeling hyperparameters can include e.g. the number of layers, the number of neurons in each layer or their activation functions.

(b) Optimisation

On top of its specific modelling approach, NN also stands out of other ML methods for its particular parameter optimisation procedure. Indeed, once the NN architecture is defined, the next step is the training of the NN. This step aims at finding the best parameters for each parameterised layer of the architecture. This procedure, as for many other ML methods involve a loss function as defined in eqs. (II.5) and (II.6). This loss describes the objective function of the NN, which is to be minimised, it encapsulates the goal of the learning task the NN ought to solve, by a comparison between the output of the NN and an expected value for this output. This loss must then be carefully crafted for an optimal task solving. An important part of the current thesis is dedicated to the crafting of such a loss function in Chapter IV, thus it is relevant to state here the importance of this concept for a clear understanding of Chapter IV.
In NN, the loss function is used in a batch gradient descent optimisation procedure. Minimising a loss function $\ell_{\mathcal{D}_{train}}(\theta_{|\theta_h})$ depending on some parameters $\theta_{|\theta_h}$ in mathematics can be thought as trivial, if $\ell_{\mathcal{D}_{train}}(\theta_{|\theta_h})$ is differentiable and convex. It only requires to find $\theta_{|\theta_h}$ values for which the gradient of $\ell_{\mathcal{D}_{train}}(\theta_{|\theta_h})$ is null, and compare the loss values for each of these parameters values to

find the general minimum of $\ell_{\mathcal{D}_{train}}(\theta)$. However, when the number of parameters increase, as well as the size of $\mathcal{D}_{train}$ this procedure becomes computationally infeasible, plenty of local minimums can exist, as well as saddle points and local maximums. Additionally, possible parameters values combinations are infinite. This is where batch gradient descent comes into play. This consists in repeatedly applying the following procedure:

1. Evaluate the loss on a randomly sampled subset, also called batch, $\mathcal{D}_{batch}$ of $\mathcal{D}_{train}$, with the current combination of parameters values $\widehat{\theta}_{\theta_h}$
2. Compute the gradient $\nabla_{\theta_{|\theta_h}}[\ell_{\mathcal{D}_{batch}}](\widehat{\theta}_{\theta_h})$ relatively to each parameter in $\theta_{|\theta_h}$
3. Update each parameter in $\theta_{|\theta_h}$ by slightly changing their value in $\widehat{\theta}_{\theta_h}$ in the direction that decrease the loss, indicated for each parameter by the opposite of gradient $\nabla_{\theta_{|\theta_h}}[\ell_{\mathcal{D}_{train}}](\widehat{\theta_{|\theta_h}})$.

The amplitude of the change in the loss decrease direction is called the learning rate r, this learning rate which evolves during the training is controlled by a so-called optimiser. Different optimisers specially crafted for NN exist; this thesis does not focus on the choice of such optimiser and rather consistently uses an optimiser that proved efficient in NN: the adaptive estimation of first-order and second-order moments **Adam** [88]. This optimiser is only sensitive to the initial learning rate chosen r which is an optimisation hyperparameter. The size of the batches $n_B = |\mathcal{D}_{batch}|$ is also an optimisation hyperparameter.
One of the reason why NN are very efficient is linked to its layered architecture. This latter lead to a mathematical function that is a nested composition of simple differentiable functions as shown in Eq. (II.32). The activation functions are indeed required to be fully differentiable. This, first enable, by using simple non-linear activation functions (e.g. sigmoid), to approximate complex functions and thus leads to the NN ability of simulating particular complex phenomenons. Secondly, this nested composition of function renders the update of parameters by stochastic gradient descent very efficient and therefore allow for a model with a huge amount of parameters. Indeed by using the chain rule $(f \circ g(x))' = f' \circ g(x)g'(x)$, and by updating the layers parameters sequentially and starting with the last one in the architecture, one can easily compute the loss gradient relative to a specific layer's parameters. It only requires to focus on computing the gradient with respect to the current layer's parameters to have $f' \circ g(x)$ and multiplying this results with $g'(x)$ the derivative of the next layer in the architecture, i.e. the layer previously updated and for which the derivative has already been computed. This parameter update process is called back-propagation and is the classical way of applying the batch gradient descent during NN training.

II–3 Conclusion

To summarise, machine learning methods for solving the specific problems defined in Subsection II–2.3 are combinations of modelling and optimisation techniques. The optimisation objective function is determined by both the model and the learning task, while solely the model output is determined by the learning task. The model is also partially determined by the context of the application and more particularly the data it needs to process. In this work, the type of data of interest are MTS monitored from sensors on complex physical systems that experience degradation, and necessarily the models and optimisation tools that are explored, are determined by this context, this holds

true for the presented overview of modelling and optimisation methods presented in this chapter. Moreover, in Subsection II–1.2, several challenges linked to the context of interest in this work were emphasised: uncertainty quantification, interdependence of MTS variables as well as their time lags and non-stationarity of the MTS under study.

The context of this thesis work and the theoretical foundation being now presented, the next chapter will focus on the overview of the literature, guided by the two research statement defined in Section I–2.

III

Overview of health indicators in PHM: construction methods and uses

III–1 Review of HI construction methods

As explained previously in Section I–2, the particular interest of this work is to exploit MTS and related methods of prognostic and health management (PHM) in a health assessment centric manner. That is, considering the task of constructing a reliable and robust health indicator (HI) able to precisely depict the degradation trajectory as a core task of PHM from which all the other tasks of PHM could be derived. With these considerations, this chapter aims at giving an overview, as exhaustive as possible, of how the task of health assessment is defined, the current methods used to solve it, and how it is connected to other PHM tasks in the literature. In the first section, a precise definition of health indicators is given, as well as the properties that are expected for a health indicator to be efficient, see III–1.1. Then III–1.2 gives a review of existing methods for health assessment. Subsequently III–2 gives a review of methods that use health assessment as a pre-task for RUL prognostic.

III–1.1 HI properties: what is a good HI?

In PHM, a health indicator is defined as a quantitative metric that assesses the system's operational status, potential degradations, or deviations from the desired performance. It takes the general form of a real number value that is indexed and evolves through time. At each time step, a new estimation of the degradation is thus given by a HI [98]. Considering the entire lifetime of a system k under study, a HI is thus a time series that takes a value $HI^{(k)}(0) = HI_0^{(k)}$ at time $t = 0$ and a value $HI^{(k)}(T^{(k)}) = HI_{T^{(k)}}^{(k)}$ at failure time $t = T^{(k)}$.
Now, to be informative, this HI needs as much as possible to satisfy some properties. Indeed, any time series monitored from a system cannot be considered as a HI.

A first proposal of general properties that a HI must satisfies has been given by [34]. In their works, the authors propose the three following properties: monotonicity, prognosability and trendability.
Monotonicity reflects the fact that a system degradation is considered to not experience self-healing without any human interventions. The HI must then continuously evolve in the same direction, i.e. increase or decrease with relation to time. This can of course be discussed, for instance as in the case of batteries that can experience short term self-healing when it is not used [210],

this latter example is given by the authors of [34] to moderate the necessity of absolute monotonicity. The same idea could hold for different devices that can experience overheating, for instance: under a hypothesis of continuous operating condition, a device close to fatal overheating due to the operation can be considered as highly degraded, while if a pause in the operation is performed the device is actually cooling and self-healing. Nevertheless, under a hypothesis of relatively constant operating condition, monotonous degradations of any system are an observable necessity, and thus it should be reflected in a HI that is monotonous. As a matter of fact, we could not find a single publication on the topic of HI construction that measures HI performance without taking into account the property of monotonicity. It can therefore be considered as the most important property of a HI.

The second proposed property in [34] is **prognosability**. The authors define it as the variance of the critical degradation values in a population of systems. Indeed, a wide spread of values for critical degradations among different instances of the system under study renders the task of extrapolating the HI to failure more difficult. This property is therefore crucial in a context where the HI is used as a pre-task for RUL prognostic, hence its name, prognosability. But even on a purely theoretic ground, having a consistent range of values a HI can take from perfectly healthy to critical degradation is mandatory for a HI to be informative.

Finally, the third property proposed in [34] is **trendability**, and is defined as the degree to which a population of systems lead to HIs that can be described by the same functional form.

These three properties have been reused numerous times in the literature [9, 108, 152], but their nomenclature and precise definitions have evolved. Consequently, some other properties, deriving from these three, have also emerged.

Prognosability as defined in [34] has indeed been derived in [8] and [138] respectively as **scale-similarity** and **failure consistency**.

For **trendability**, the term is used in a different sense as the one proposed in [34]. It has been used as a property expressing a correlation between the HI value and time, e.g. in [97]. This new use of trendability can now be commonly found, e.g. in [138], since it has been reported in that sense in a influent review on the domain [98]. This new definition of trendability becomes very close to the property of monotonicity, and may be viewed as a variation of this latter.

Finally, a last property that emerged in the literature and that is different to all the previously mentioned one is **robustness**: it expresses the absence of random fluctuation, a.k.a. noise, that might be due to e.g. the stochasticity of the degradation processes or the variation of operational conditions [98].

Due to the evolving terminology and definitions surrounding the properties of HI in the literature, this work has made certain choices for clarification purposes. **Monotonicity** usage and definition is consistent through the literature and is thus kept as in the definition of [34, 9, 108, 152, 98]. The term **prognosability** in [34] on the contrary does not precisely reflect the idea it

attempts to define, thus the term failure consistency proposed in [138] has been preferred. The term trendability as used in [34] for defining the similarity of functional form is found to be misleading, as it is now used in the sense reported in [98] and the term **prognosability** is thus assigned for this property. Indeed, the prognostics of future HI values of a system's instance is rendered easier if all instances have similar HIs sharing the same functional form. **Prognosability** hence seems a more adequate name for this property than for the previous one. Finally, robustness was kept untouched and is used as presented above.

To resume, here are the definitions given for each of the retained HI properties in this work:

Monotonicity defines the consistency of evolution of the HI in the same direction. More precisely, in this work, the HI of an instance k of a system is seen as a degradation value that evolves from an initial healthy value of $HI_0^{(k)} = 0$ and that reach a final failure value higher than $HI_{T^{(k)}}^{(k)} > 0$. The monotonicity is therefore expressed as the monotonous increase of the HI.

Failure consistency expresses the fact that a HI construction model applied to a specific system, produces, for each instance of the system, a HI whose values at failure time are similar.

Prognosability expresses the fact that a HI construction model applied to a specific system, produces, for each instance of the system, a HI that can be predicted using the HI from the other instances of the system as examples to train the prediction model. That is, that every instance produce HIs with the same functional form.

Robustness expresses the absence of random fluctuation, a.k.a. noise, that might be due to e.g. the stochasticity of the degradation processes or the variation of operational conditions.

Additionally, a summary of the terminology used in the literature and used in the present work is given in Table III.1, as well as the mathematical definitions used in this work for these properties. Some further details must be expressed on the mathematical definitions of these properties in Table III.1:

Mon in the **monotonicity** criteria mathematical definition which is the same as the one proposed in [34], $N_{dHI_t>0}$ and $N_{dHI_t<0}$ are the number of times the difference between two consecutive HI values are positive and negative, respectively. Additionally, one should note that computing this indicator on the raw HI can be misleading, especially if the HI is noisy. Hence, it is advised first to denoise the HI or compute the dHI_t on more than one step, i.e. $dHI_t = HI(t) - HI(t - \Delta t)$ with $\Delta t > 1$. In this work, we use $\Delta t \approx 0.05T$ where T is the total length of the HI. A value of 1 or -1 indicates that $HI(t)$ is strictly monotonously increasing or decreasing, respectively. The closer to zero, the less monotonous the HI.

FC the proposed mathematical definition of **failure consistency** in this work is an original one, directly related to the prognosis task. It stands for the mean amount of time $t_{Th}^{(k)}$ before a HI curve $HI^{(k)}$ reach the failure threshold value Th relative to the HI total amount of time $T^{(k)}$. The threshold value Th is here set as the minimum, over all the n computed HIs, of the maxima values of each HI. If, on one hand, FC is close to one, it means all HI curves reach a common threshold at the end of life and are therefore consistent in terms of failure. On the other hand, if FC is close to zero, then a common threshold reached by all the HI curves at their end of life cannot be found. An illustration of the computation procedure can be found in Figure III.1

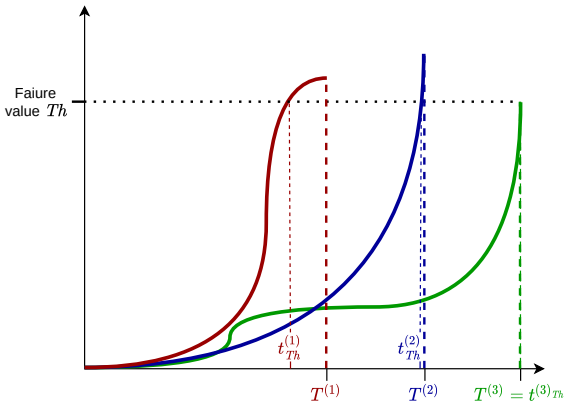


Figure III.1: Illustration for the **failure consistency** computation

Pro in the **prognosability** criteria mathematical definition, also used in e.g. [9], $\{\overrightarrow{HI^{(k)}}\}_{k\in\mathcal{K}}$ are vector projections of $\{HI^{(k)}(t)\}_{k\in\mathcal{K};\, t\in\mathcal{T}_k(T^{(k)})}$ in a $\mathbb{R}^M$ space, where $M = \max_{k\in\mathcal{K}}(T^{(k)})$ is the number of time step of the longest HI. All the HI trajectories are linearly projected so that the M^{th} value of their projection is equal to their HI value at failure time $HI^{(k)}(T^{(k)})$. Because all the HI except the longest one have less than M time steps, their projection's values are linearly interpolated. If Pro is close to one, then the HI curves are highly correlated, and the prognosis is made easier. In contrast, if Pro is close to zero, then at least one curve significantly differs from the others and leads to a much more complex prognosis task. An illustration of the computation procedure can be found in Figure III.2

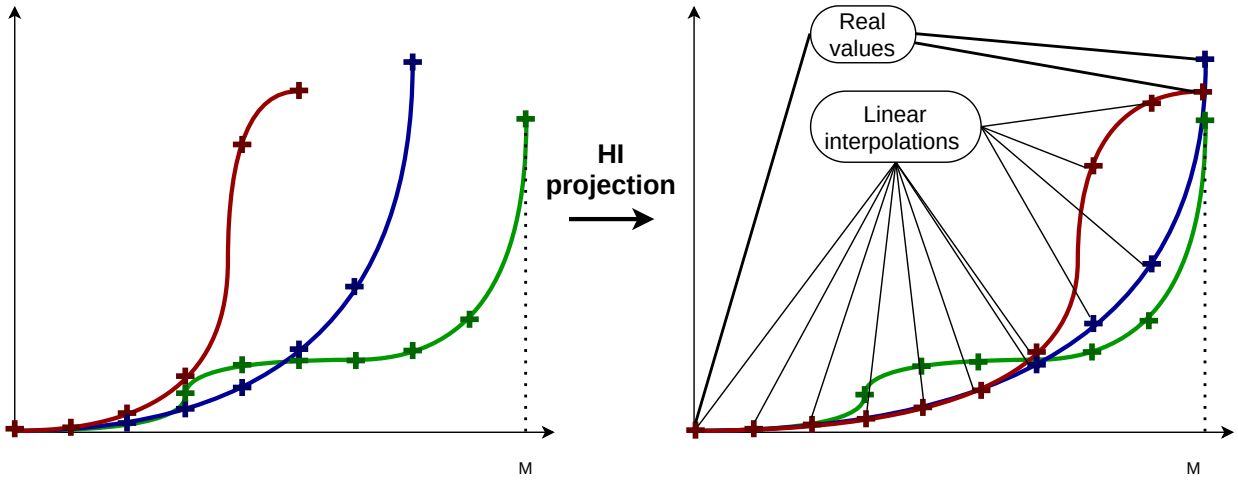


Figure III.2: Illustration for the **prognosability** computation

Rob in the **robustness** criteria mathematical definition, also used in [29, 223, 152], $HI^{(k)}(t)$ is decomposed into a trend $T_{HI^{(k)}}(t)$ and a residual $R_{HI^{(k)}}(t)$. The locally estimated scatter plot smoothing (LOESS) [33] is used for extracting the trend, $T_{HI^{(k)}}(t)$ while the residuals are the differences between the HI and its trend. To be performed, LOESS needs one parameter which is the window length l_w defining the localised subset of $HI^{(k)}(t)$ values needed to compute $T_{HI^{(k)}}(t)$ for all t, similarly to a sliding window operation. In this work, we used a window

size of five percent of the total $HI^{(k)}$ length, $wl = 0.05I^{(k)}$ where $T^{(k)}$ is the time of failure for the k^{th} instance of the studied system. A value of zero indicates a perfect robustness of $HI^{(k)}$, while an increasing value indicates a decreasing robustness. Furthermore, a value of e.g. 0.1 would indicate that the residuals account for 10% of the HI values on average.

term used in [34]	different term in the literature	term used in this work	mathematical formulation in this work
monotonicity	monotonicity [9, 108, 152, 98]	**monotonicity**	$\mathbf{Mon}(HI^{(k)}) = \frac{N_{dHI_t^{(k)}>0}}{N_{dHI_t^{(k)}}} - \frac{N_{dHI_t^{(k)}<0}}{N_{dHI_t^{(k)}}}$ $dHI_t^{(k)} = HI(t) - HI(t-1)$
prognosability	scale-similarity [8], failure consistency [138]	**failure consistency**	$\mathbf{FC}(\{HI^{(k)}\}_{k\in\mathcal{K}}) = \frac{1}{n}\sum_{k=1}^{n} \frac{t_{Th}^{(k)}}{T^{(k)}}$ where $t_{Th}^{(k)} : HI^{(k)}(t) \leq Th\,,\, t < t_{Th}^{(k)}$ and $Th = \min_k(\{\max_t(HI^{(k)}(t))\}_{k\in\mathcal{K}})$
trendability	-	**prognosability**	$\mathbf{Pro}(\{HI^{(k)}\}_{k=1}^{n}) = \min_{\substack{i\neq j \\ i,j\in[1,n]}} (r_{ij}(HI^{(i)}, HI^{(j)}))$ $r_{ij}(HI^{(i)}, HI^{(j)}) = \text{Corr}[\overrightarrow{HI^{(i)}}, \overrightarrow{HI^{(j)}}]$
-	trendability [97, 138, 98]	-	-
-	robustness [98]	**robustness**	$\mathbf{Rob}(HI^{(k)}) = \frac{1}{T^{(k)}+1}\sum_{t=0}^{T^{(k)}} \frac{\lvert R_{HI^{(k)}}(t)\rvert}{T_{HI^{(k)}}(t)}$ $HI^{(k)}(t) = T_{HI^{(k)}}(t) + R_{HI^{(k)}}(t)$

Table III.1: The different terminologies of HI properties in the literature and the chosen ones in the present work, associated with their mathematical definitions

If measurable signals comply with these properties, they can be directly used as HI. They are called Physical Health Indicator (PHI). Sometimes statistical features of measurable signals can comply with HI properties and therefore can also be considered as PHI. Usually, finding a robust PHI necessitates an expert knowledge on the system's physics under study, it also has the drawback of being extremely context-dependent. In most cases, anyway, no single physical or measurable quantity of interest (QoI) is known to represent the degradation state of a complex component operated under various conditions. One should therefore construct a Virtual Health Indicator (VHI). That is, a quantity obtained via a data driven selection and aggregation of multiple monitored

signals. The interest in this work is focused on this second type of HI for their usefulness in the study of complex physical systems and for their potential multi-context applicability. Hence, from now, we use HI to denote VHI.

III–1.2 Construction methods

III–1.2.1 Time series preprocessing

Whilst constructing a HI of a system from multiple signals monitored on it, it is often required to first preprocess these signals. In this work, only operations that are applied to individual signals, i.e. to any variable $j \in 1, \ldots, d$ of a MTS, and that do not combine them, are considered to be preprocessing. Operations where multiple signals are aggregated or combined are considered as being part of the HI construction model itself and not a preprocessing step. In some cases raw time series do not require any preprocessing, e.g. [208, 124, 87, 103, 220, 60], but generally three recurrent preprocessing operations can be found in the literature related to HI construction methods: **feature extraction**, **denoising** and **filtering**. Most often, at least one of these three preprocessing tasks is necessary before the construction of a HI. A summary of which preprocessing operations are performed in some influent HI based approaches is given in Table III.2.

Feature extraction	Denoising	Filtering	References
-	-	-	[208, 124, 87, 103, 220, 60]
-	-	1	[180, 111, 131]
1	-	-	[99, 153, 218]
1	2	-	[192]
-	1	-	[29, 28]
1	-	2	[8, 62, 148, 9, 97, 68, 138]

Table III.2: Preprocessing steps and their order of application in the HI construction methods.

Feature extraction encompasses any transformation of raw signals that produces new and more informative signals, facilitating subsequent HI learning. These operations include the evaluation of common statistical and temporal measures on given time windows of length l, e.g. mean, standard deviation, kurtosis, and skewness [62, 97, 8, 150, 148, 138]. This type of feature extraction is represented on Figure III.3 for illustration purpose and is the most used for its simplicity of implementation. Time windows are either defined by the nature of the recorded MTS in the case of MTS with subseries-uniform sampling, or empirically constructed by a sliding window operation, with Δ the sliding step, in the case of MTS with uniform sampling, as shown on figs. III.3 to III.5. More complex measures on the frequency domain, e.g. centre frequency, mean frequency or total energy and energy ratio of specific frequency bands can also be used [150, 97, 62]. These extracted features can be qualified as frequential and are obtained in practice by a two-step procedure depicted on Figure III.4. First, on similar time windows as previously explained, is applied to the signal, either an operation which transforms the signal in the frequency domain, i.e. a Fourier transform, or an operation which decomposes the signal into several components. A trend-seasonal-residual decomposition, an Empirical mode decomposition (EMD), discrete wavelet transform (DWT), or

any other operation that consist in filtering some frequency bands of the signal can be considered. Then, specific measures related to the new representation domain of the signal are applied, e.g. centre-frequency or total energy of frequency bands.
Finally, the transformation of the signals mentioned before, or more complex ones, is also included in feature extraction. The followings can be cited in that sense: the decomposition of signals in several intrinsic modes by means of EMD [47, 100], the Hilbert-Huang transform [181] or discrete wavelet transform [62, 40], transformations of signals in the frequency domain, e.g. the Fourier transform [146, 219], or in the time-frequency domain e.g. by means of continuous wavelet transform[218] or short-time Fourier transform [197, 217, 227, 234]. This type of extraction is depicted on Figure III.5.

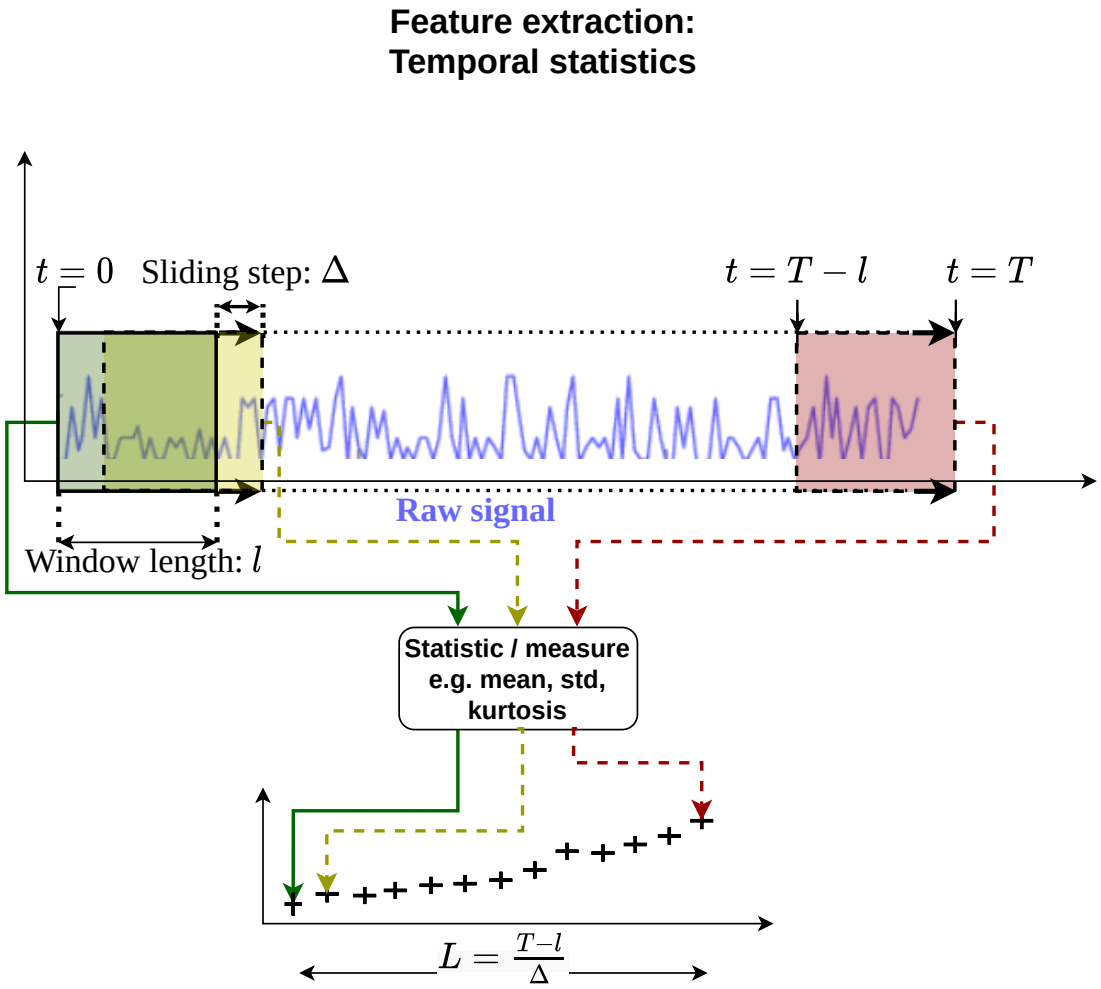


Figure III.3: **Feature extraction** with classical temporal statistics

Filtering or **feature selection** is another preprocessing practice applied on raw signals or on extracted features, whose purpose is to only keep signals or features that comply with properties required for a HI, i.e. **monotonicity**, **trendability**, **prognosability**, **robustness** or **failure consistency**. The underlying idea is then to obtain a HI based on signals or features which satisfied the above-mentioned properties of a HI. For instance, consistently used **feature selection** criteria are the **monotonicity** or **trendability** scores, whose high values lead to selection of the signal

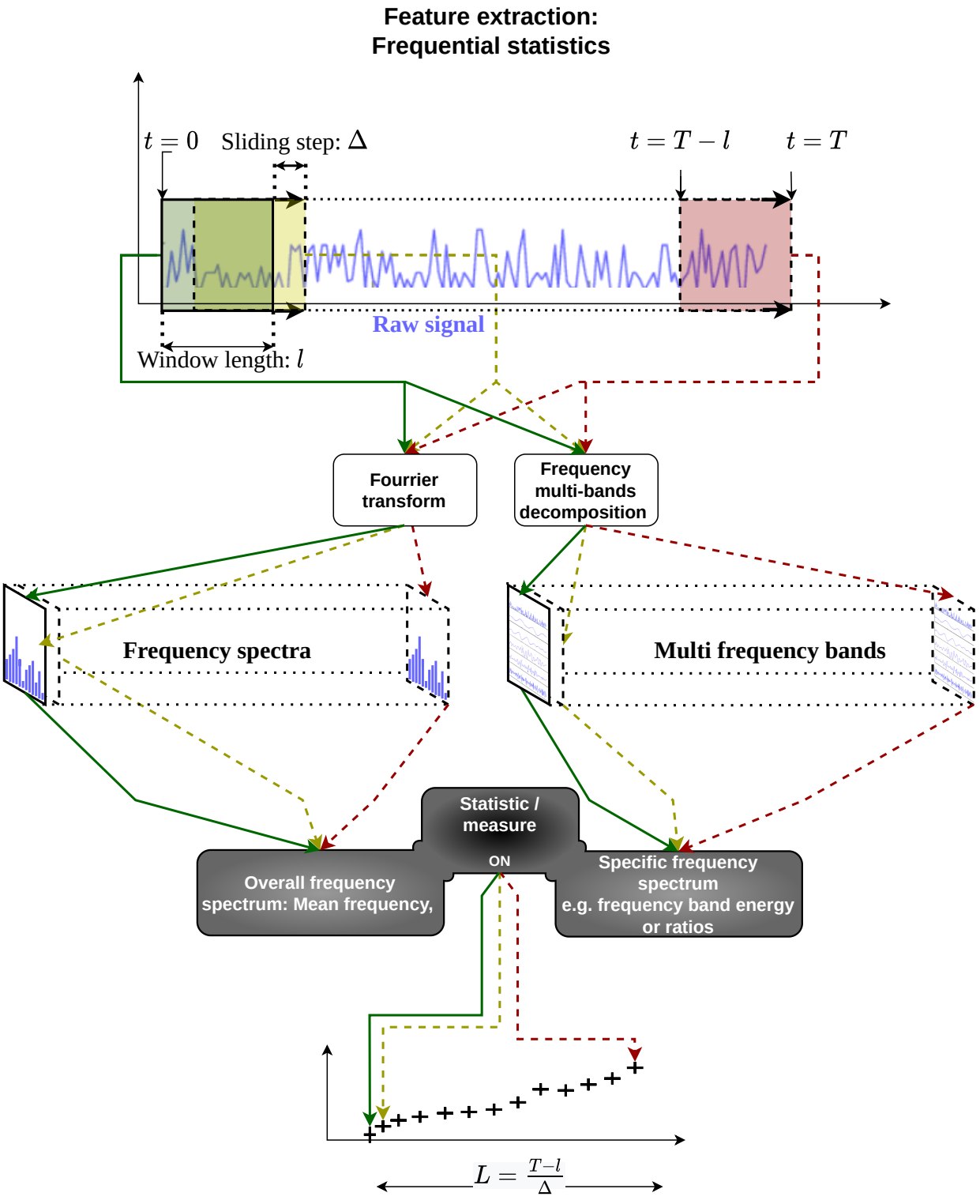


Figure III.4: **Feature extraction** with frequential statistics

or feature [180, 8, 62, 148, 9, 97, 68, 138]. If these two scores compute two different things like mentioned in Subsection III–1.1, they both evaluate a similar property which is the monotonicity. Other HI properties are also used to select features during this preprocessing step, that is the case for robustness, e.g. in [8, 138], or prognosability and failure consistency, e.g. in [9, 138]. Never-

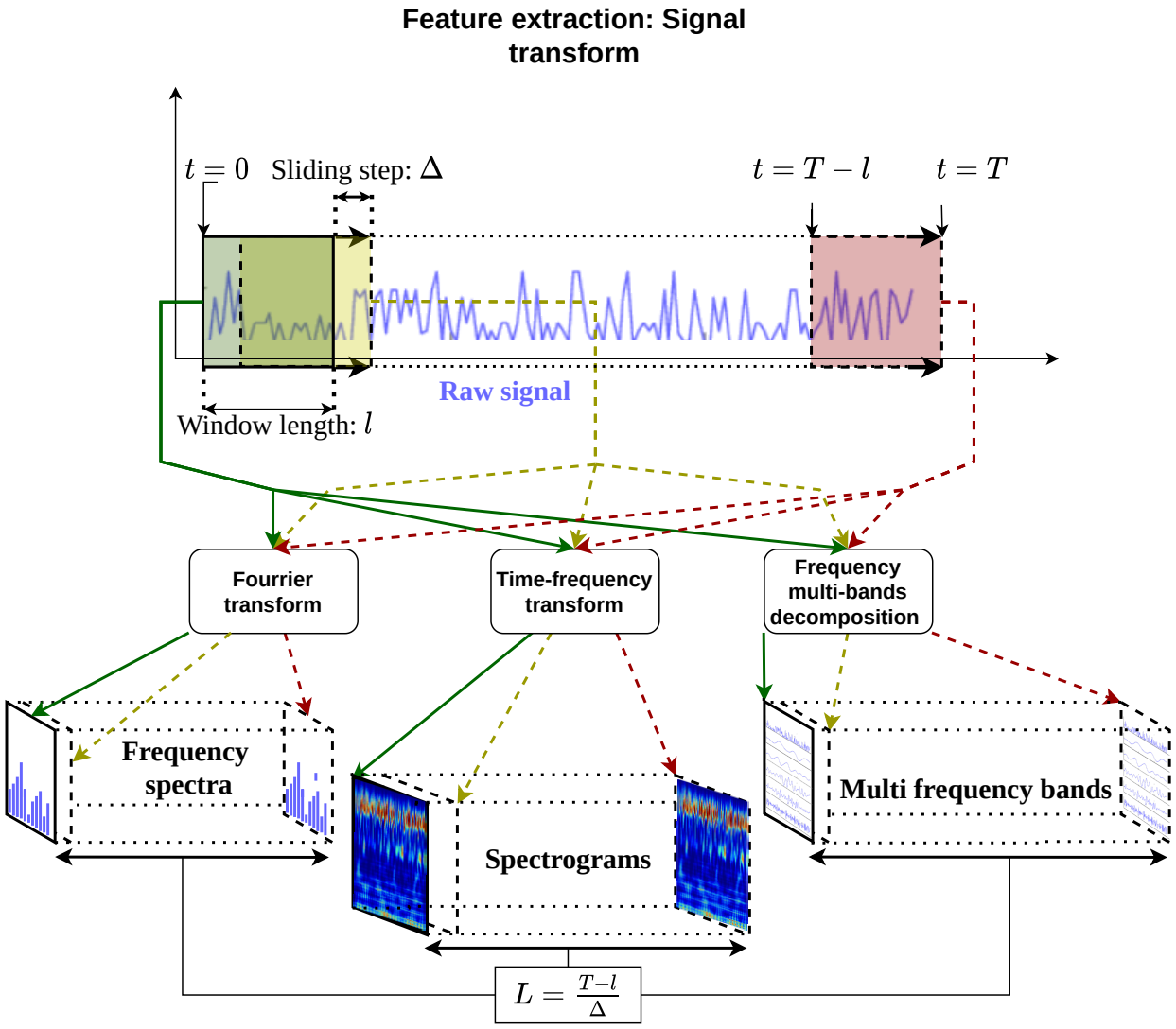


Figure III.5: **Feature extraction** with transformations of signal subseries

theless, the monotonicity property is always accounted for and is therefore acknowledged by the literature as the most important or obvious property a HI must satisfy.

Finally, the last preprocessing practice consists in **denoising** or **smoothing** the raw signals or extracted features [29, 28, 192]. The objective is to remove the measurement noise or outliers in the signals under study, therefore increasing the **robustness** of the HI obtained subsequently. Similarly, this practice can also be used as a "post-processing" step on the final HI, to make it more robust to noise.

III–1.2.2 HI construction

After the preprocessing step, the obtained signals need to be merged into a single HI, combining partial and diverse information about the equipment degradation state contained in individual signals or extracted features. This step is the core of HI construction. The existing approaches for

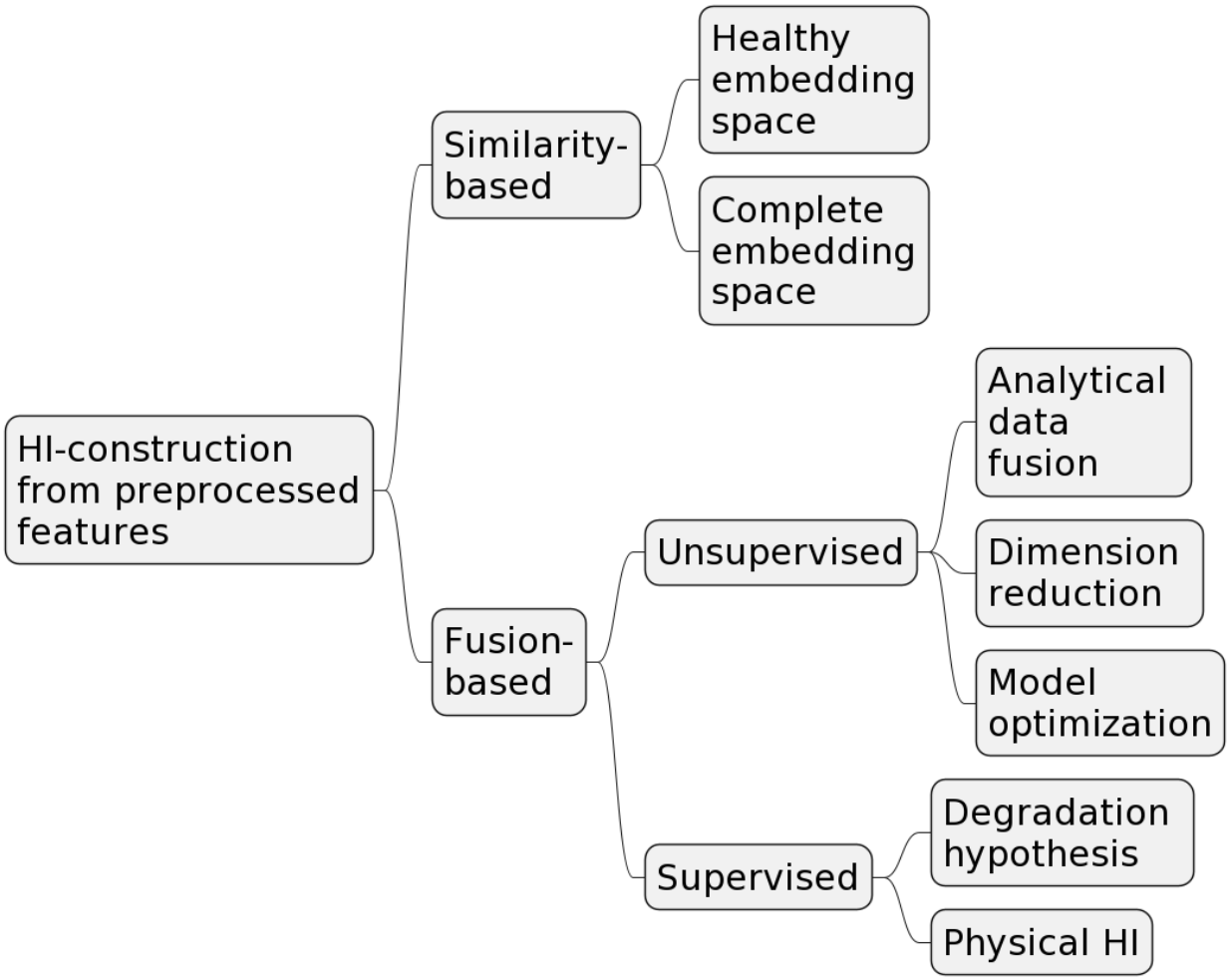


Figure III.6: Classification of HI construction methods.

this operation can be classified according to Figure III.6. Two distinct strategies are available: **fusion-based** and **similarity-based** strategies. These two different paradigms are illustrated on Figure III.7

(a) Fusion-based approach

A **fusion-based** strategy consists in learning a single function f which maps the data of each system's instance k, with $k = 1, \dots, n$, resulting from the preprocessing step, noted $X_t^{(k)} = \left\{X^{(k)}(t)\right\}_{t \in \mathcal{T}_k(T^{(k)})}$, into a univariate signal $HI^{(k)}(t) = f(X^{(k)}(t)) \in \mathbb{R}$. $X_t^{(k)}$ can be of different natures depending on the preprocessing procedure. If the raw data are MTS with subseries-uniform sampling and that preprocessing is deprived of any feature extraction, then $X_t^{(k)}$ are also MTS with subseries-uniform sampling. If the preprocessing includes feature extraction of temporal or frequential statistics, then $X_t^{(k)}$ are uniform MTS. Finally, if the feature extraction consists of a signal transform, $X_t^{(k)}$ are time-indexed series of the new representation of the signals resulting from this transform.

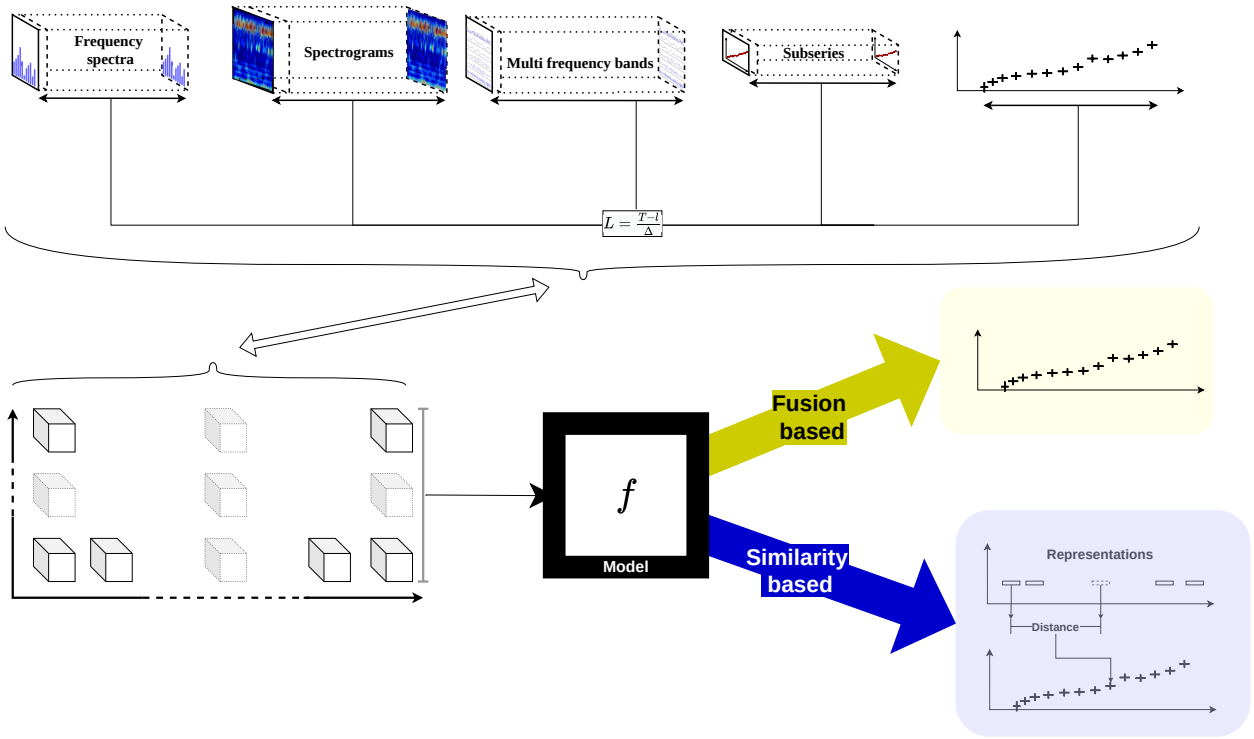


Figure III.7: Representation of the two different HI construction paradigms: fusion-based and similarity-based.

The **fusion-based** approach comes in two different flavours, quite as similarly as in machine learning, with **supervised** and **unsupervised** techniques.

Supervised refers to approaches whose mapping function f is optimised to produce $HI^{(k)}(t)$ as close a possible to a theoretical HI $\widehat{HI^{(k)}}(t)$ for any given $X^{(k)}(t)$. For this approach, $\widehat{HI^{(k)}}(t)$ needs to be predefined either by a known PHI or a HI degradation hypothesis, e.g. a linear HI with $\widehat{HI^{(k)}}(0) = 0$ to $\widehat{HI^{(k)}}(T_k)) = 1$. A convolutional neural network (CNN) in [218] and a recurrent neural network (RNN) in [62] are e.g. trained with an assumed linear HI. In the context of LI-Ion batteries, [87, 103, 77] resort to an existing and robust PHI, the battery capacity, as supervisor of their mapping model respectively being: a perceptron, a long-short time memory (LSTM) neural network (NN) and a Gaussian process. The main drawback of both approaches is the need for a PHI as a supervisor, which may not exist, or an assumption about the evolution of degradation which may not necessarily be consistent with the real but unknown physical degradation process.

To alleviate this shortcoming, **unsupervised fusion-based** methods have been developed and can further be grouped into three classes: **analytical data fusion**, **dimension reduction** and **model optimisation**.

The most common approach is **analytical data fusion**, where the fusion model $f(X(t)^{(k)})$ is analytically designed from expert knowledge. [99] e.g. proposes a pair-wise linear fusion whose weights are based on relative entropy, while [8] uses an adaptive linear fusion with dynamic weights evolving with time according to inter-features distances.

Dimension reduction techniques like principal component analysis (PCA) [131] or linear local embedding (LLE) [68] are also used as fusion models by applying them to the preprocessed features in order to reduce their dimension to one.

Analytical data fusion and **dimension reduction** based approaches are not mathematically driven to produce a monotonous HI, or to satisfy any other HI properties. They therefore rely on the preprocessing step, especially the **feature selection**, to ensure that the HI satisfies some expected properties, e.g. monotonicity.

By contrast, other **unsupervised fusion-based** strategies that enable constraints enforcement on the HI, have thus been developed. They are often referred to as **optimisation models**. They consist in optimising the parameters of their fusion model given one or several HI performance indicators as optimisation objectives, i.e. monotonicity or failure-consistency. [111, 180] use e.g. a linear fusion model optimised for maximising the width of the entire HI value range and minimising the HI polynomial fitting error as well as the width of the end-of-life HI value range. However, most works on the topic focus on optimising on HI's properties such as the **scale-similarity** of HI's end values but no work is found on incorporating **monotonicity**, which can be explained by the difficulty to integrate ranking constraints between data samples in a convex, linear or quadratic optimisation problem. More recently, Genetic algorithms (GA) have been proposed as a different approach of HI fusion model optimisation with complex objective functions. They allow one to explore a more diverse set of fusion models, thanks to their ability to define a function space, where not only numerical parameters vary, but also the candidate non-linear functions used in the model itself. Most importantly, they allow the definition of complex objective functions, that only require low computational complexity but no property on its derivative, as it can be the case for gradient based optimisation. These objective functions can then easily integrate constraints of monotonicity or failure-consistency, for instance. [208] proposes e.g. a generic approach using GA to create a HI construction model and [138] proposes GA for automatically extracting the relevant features from the measured signals as well as for the HI construction model, both approaches successfully incorporated **monotonicity** constraints in the objective functions of their GA.

(b) Similarity-based approach

As an alternative to **fusion-based** strategies, the HI construction can be set up in a completely different way, by using a **similarity-based approach**. It consists, firstly, in a representation learning problem, i.e. learning a projection of the data obtained after the preprocessing step, $X_t^{(k)} = \left\{X^{(k)}(t)\right\}_{t\in\mathcal{T}_k(T^{(k)})}$, in an embedding space, a.k.a. latent space. As mentioned above, $X^{(k)}(t)$ can be vectors, MTS subseries of dimension d or they can be sets of d representation of the original signals. In any case, the goal of representation learning is to embed them in a space of

dimension e which must be lower than the original dimension and in which these subsamples are separable in terms of their degradation level. The HI is then constructed based on the distance between data samples taken at different times, projected in this representation space. Two different approaches exist: learning the representation space, either only on healthy data, $t \in \{0, \dots, \tau^{(k)}\}$ with $\tau^{(k)} << T^{(k)}$, or on the entire dataset, $t \in \{0, \dots, T^{(k)}\}$. In the first case, the HI will be the distance between a data sample projection at time t and the closest healthy sample projection. In the second case, the HI will be the distance between a sample projection at time t and the sample projection of the same component at time $t = 0$. In [97] the authors e.g. use a self-organising map as the representation learning model and train it only on healthy data while [220] and [60], e.g., both use RNN based auto-encoders as representation learning model and train them on the entire dataset. Alternatively, in [9] the authors use an auto-associative kernel regression (AAKR) as their representation model, and train it only on healthy data. The drawback of these methods are again the absence of monotonicity constraint in the model. They only rely on feature filtering for satisfying the monotonic property of the obtained HI.

If **fusion-based** strategies have been developed to the point of integrating the monotonicity constraint or any other constraint relative to the HI properties, thanks to the recent development of genetic algorithm approaches, few works have been found on **similarity-based** approaches in that direction. Currently, only the works of [150] attempt to tackle this issue. However, recent advances in deep learning models, such as in representation learning, similarity learning or contrastive learning could be beneficial to similarity based health indicator construction model. As a matter of fact, the authors of [150] use recent advances in deep representation learning, more precisely variational auto-encoder (VAE) to incorporate, in the loss function of a NN optimisation, a constraint of monotonicity. More precisely, a reconstruction self-supervised setting was used to encode MTS subseries in a unidimensional latent space of a VAE. The monotonicity constraint is then applied on the latent space during the training. Each subseries is fed to the VAE in parallel with its closest subseries in the past, the former's latent value is then constrained to be higher than the former's one. The unidimensional latent space projections of the MTS subseries is thus used as the HI. Inspired by this work, and by recent works of contrastive learning, i.e. the representation learning of time series by means of siamese neural networks (SNN) and triplet loss proposed in [50], we thus propose in Chapter IV a new methodology for HI construction based on SNN and triplet loss with the incorporation of a monotonicity constraint.
Moreover, the use of similarity learning tools, enables the learning of meaningful representation of the raw signals that could potentially be used for the other predictive maintenance tasks, i.e. RUL prognosis, failure mode classification and anomaly detection.

III–2 Review on HI-based RUL prognostics

An overview of the current methods, and their limitations, for HI construction has been given in the previous section. To pursue our examination of PHM data analysis methods in a health assessment centric way, this section focuses on a PHM task that is notably facilitated by the construction of an HI: the RUL prognostic. Indeed, although an HI can be a valuable outcome for PHM in itself, the idea of constructing one first appeared in the literature for solving the RUL prediction task [114]. Consequently, this section will give an outline of RUL methods which rely on a health index. But

firstly, it is important to recall that also non-HI based RUL prediction approaches also exist, but that these latter are not in the scope of the current work. Nevertheless, the two following paragraph aim at briefly presenting both paradigms.

(a) Non-HI-based RUL prognosis

Before diving in the HI-based RUL estimation methods, this paragraph briefly discusses the methods that are non-HI-based, often referred to as direct RUL prediction.
RUL prognosis has originally been solved via the direct monitoring of a degradation measure (PHI) or the construction of a VHI. A more precise description of HI-based RUL prognosis is given in the next paragraph. Nevertheless, recently, the breakthrough of deep learning, also enables to construct complex model that can map the current monitoring data of an instance to its RUL prediction. The advantages of this non-HI-based approach is of course the simplicity of setting up the model because it circumvents the construction of HI models, forecasting models and threshold estimations. The idea is to map, for any time t, the last few measurements of monitored signals (possibly preprocessed) to the current time until failure. This also gives the advantage of not needing the full history of measurements for making a prediction, which can be the case in certain HI-based approaches. Many architectures and techniques of NN have been applied to learn such mappings, can be cited e.g., multi-layer perceptrons [211], convolutional neural networks [67, 105, 199, 235], recurrent neural networks [67, 199] and attention mechanisms [23, 226]. The primary disadvantage of this approach is the requirement for a substantial quantity of data – specifically, a considerable number of instances of the system under investigation that have been monitored and ran to failure. Additionally, disregarding the HI modelling step, if convenient for the sake of simplicity, also increases the opacity of the model, which is already at a high level considering the ML method used, i.e. deep learning.

(b) HI-based RUL prognosis

The subject of interest in this work is the HI-based RUL estimation methods. The latter are the most numerous in the literature, for the main reason that the non-HI-based methods have appeared only recently, as mentioned in the previous paragraph. HI-based methods have the advantage of going through the process of HI construction, which can in itself give important informations about the different degradation dynamics that occur in the system under study. They are therefore more explainable than the direct deep learning based approaches. Other advantages can be awarded to this approach, but first a brief explanation on the principles of HI based RUL prognosis is necessary.

1. Elaborating a forecasting model, that can predict a HI trajectory given its history.

2. Proposing a failure threshold estimation model, that can give a threshold value above which the system's instance under study is considered failed.

The present section gives an overview of the different methods existing in the literature to address both challenges.

III–2.1 HI forecasting

As mentioned just before, once the HI model is available, the HI trajectories of observed instances ran to failures can be obtained, denoted as:

$$\mathcal{HI} = \{HI^{(k)}(t)\}_{k\in\mathcal{K},\, t\in\mathcal{T}_k(T^{(k)})}$$

Additionally, the HI model can give HI values for any instance κ observed until a given time τ. The goal is now to build a model able to forecast the evolution of a specific HI trajectory given its past values.

The forecasting task can be formalised as follows: for any instance κ, whose incomplete HI trajectory is $\{HI^{(\kappa)}(t)\}_{t\in\mathcal{T}_k(\tau)}$ with $\mathcal{T}_k(\tau) = \{0, T_s, \ldots, \tau\}$ for any value of $0 < \tau < T^{(\kappa)}$, it is desired to find the future values of the HI trajectory $\{HI^{(\kappa)}(t)\}_{t\in\mathcal{T}_k^+(\tau)}$ with $\mathcal{T}_k^+(\tau) = \{\tau, \tau+T_s, \ldots, \tau+h\}$ for any horizon $h > 0$. Because the time of failure $T^{(\kappa)}$ is unknown, the forecasting horizon h can be viewed as infinite.

As observed in [202], two main settings have been proposed so far for this task:

Individual model In this setting, all instances are studied separately, one single model is constructed for each instance, the model parameters are then estimated only via past observations of the trajectory $\{HI^{(\kappa)}(t)\}_{t\in\mathcal{T}_k(\tau)}$.

General model In this setting, the HI of all instances are considered to share some similarities but with particular differences for each trajectory. From a statistical point of view, they can be thought as particular realisations of a same stochastic process. Therefore, both the already observed trajectories in the set $\mathcal{HI}$ and the historical values of the current trajectory $\{HI^{(\kappa)}(t)\}_{t\in\mathcal{T}_k(\tau)}$ are used to predict the future values of the latter.

The HI forecasting task has been studied with different type of approaches, which have in turn been classified given different taxonomies. The recent review on RUL prognostic [98] attempts to clarify the different type of approaches. The classification given hereafter is partially inspired from this review, but some changes can be emphasised for the reason that this review study RUL prediction approaches regardless of the fact that they are HI-based or not and also ignoring the difference between individual model en general model settings. In their review, [98] differentiate between *physics model-based approach*, *statistical model-based approach* and *artificial intelligence based approach*. In this work, we decide to classify the approaches based on their mathematical formulation of the forecasting problem rather than the precise nature of the methods used to solve it, as it has been done in [98]. Four categories are therefore used to classify HI forecasting model in this work:

- Function fitting
- Curve matching
- Machine learning
- Stochastic process

III–2.1.1 Function fitting

Function fitting models encompass models whose goal is to provide, for any particular incomplete trajectory of an instance κ until $t = \tau$, a function $\widetilde{f}_{\Theta^{(\kappa)}} : \mathbb{R} \to \mathbb{R}$, $t \mapsto \widetilde{f}_{\Theta^{(\kappa)}}(t)$ indexed on time and parameterised by $\Theta^{(\kappa)}$. The functional form of $\widetilde{f}_{\Theta^{(\kappa)}}$ is chosen based on expert-knowledge and known physical dynamics of the system under study. More precisely, these functional forms can follow some known law of degradation, e.g. Paris-Erdogan in [213, 149] or Forman law in [141] for crack-growth application. Alternatively, researchers can infer these functional forms by observing existing trajectories; for example, polynomial [4] or exponential models [124] can be implemented. For this category of HI forecasting, only an individual model setting is considered, the optimal parameters $\Theta^{(\kappa)}$ are determined by historical values of the trajectory of the instance κ, usually via least squares minimisation, as follows:

$$\Theta^{(\kappa)} = \min_{\Theta} \sum_{t=0}^{\tau} |HI^{(\kappa)}(t) - \widetilde{f}_{\Theta}(t)| \tag{III.1}$$

Once the function $\widetilde{f}_{\Theta^{(\kappa)}}$ is fitted, it is trivial to give a prediction of the HI value for any future time $t = \tau + h$, with h a potentially infinite horizon. A schematic overview of this type of approach is represented in Figure III.8.

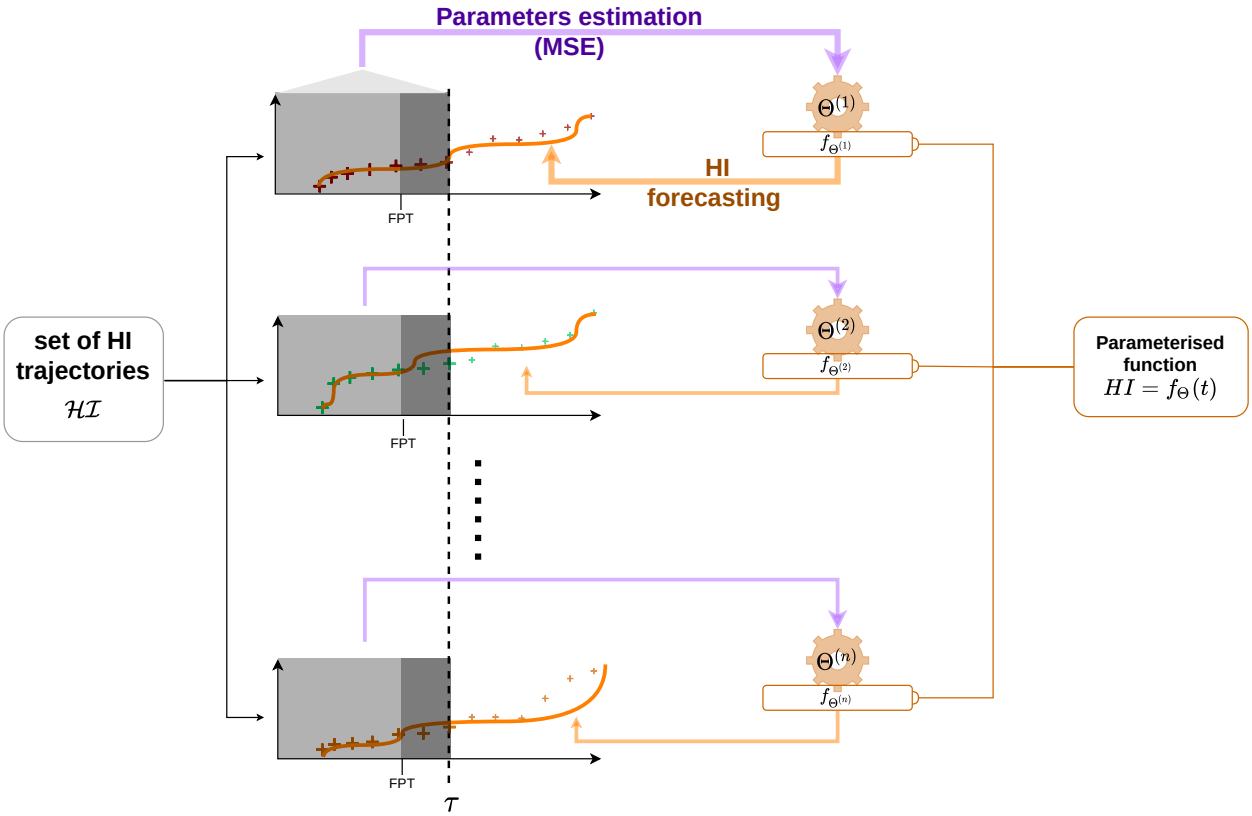


Figure III.8: Schematic view of function fitting HI forecasting.

In practice, with this approach, as for any approach in the individual model setting, a sufficient amount of known points of the trajectory must be observed before starting to ensure a correct estimation of the parameters $\Theta^{(\kappa)}$. Moreover, at the start of life of a system, the degradation

evolves slowly and is challenging to identify. A common solution to bypass this problem is to detect when a HI value indicates a significant increase in the degradation compared to past values. The time when such a HI value appear is called the **first prediction time (FPT)**, it is denoted here t_{FP} and most often defined by an arbitrary rule. For example, the authors of [4] start to predict HI values when an observed HI value increased by 10% w.r.t. an HI value measured at a predefined time lag in the past, while the authors of [102] wait for the HI to be higher than thrice the variance of previous recorded values. In this context, the approach then consists for any $\tau \geq t_{FP}$ to estimate $\Theta^{(\kappa)}$. At each time step, new measurements of the HI trajectory are known, and therefore, given that the functional form is well-fitted to the system under study, the predictions are improved. This category includes only deterministic approaches, and therefore do not enable one to quantify the uncertainties. When these latter are taken into account in this modelling approach, that is, in the model parameters or from the measurement noise, the model can be designed as a stochastic process which belong to the category defined below in Subsection III–2.1.3.

III–2.1.2 Curve matching

Curve matching approaches differ from function fitting in the sense that they rely on all the observed complete trajectories, and the more there is the better. The underlying idea is based on the comparison of each new trajectory $\{HI^{(\kappa)}(t)\}_{t=0}^{\tau}$, observed until time τ, with all the available complete trajectories in the set $\mathcal{HI}$. To do so, usually, each trajectory is only considered up to time τ and a distance or similarity measure is computed between the new trajectory and every observed ones:

$$d(\{HI^{(\kappa)}(t)\}_{t\in\mathcal{T}_\kappa(\tau)}, \{HI^{(k)}(t)\}_{t\in\mathcal{T}_k(\tau)}),\ k \in \mathcal{K} \tag{III.2}$$

Once all the observed trajectories have been compared, they are weighted based on their similarity with the new partial trajectory, to give the weights: $\{w_k\}_{k\in\mathcal{K}}$. Then a calibrated function is designed to combine and merge together the observed trajectories in $\mathcal{HI}$ according to their respective weights $\{w_k\}_{k\in\mathcal{K}}$. The result of this function can be a new trajectory or a trajectory distribution. It is interesting noting that this approach can be used without failure threshold estimation. In such situation, the RUL is directly forecasted by merging the RUL of the observed trajectories at time $t = \tau$ according to their respective weights $\{w_k\}_{k=0}^{n}$. A schematic overview of this type of approach is represented in Figure III.9.

In this category of approaches, can be cited [220] where the authors propose to compare the HI trajectories based on Euclidean distance, then the RUL is directly predicted by the best matching trajectory, which may rather be viewed as trajectory selection than combination. The authors in [121, 60] use a similar trajectory comparison technique, but rather predict the RUL as the average of all observed trajectories weighted by their respective similarity to the current trajectory. The authors of [53, 113] also resort to a weighted average of observed trajectories based on similarity, but where the similarity measure has been obtained respectively by cosine distance and by a combination of Euclidean and cosine distances. The authors in [131] use a clustering technique (K-NN) to split the observed trajectories into several clusters, then the new partial trajectory is assigned to one cluster, and the trajectories of this latter are used for prediction. [154] also recently proposed a method called RULClipper in case of noisy HIs. It first consists in determining the envelope of each trajectory, the similarity of two trajectories is then based on the area of intersection of their respective envelopes.

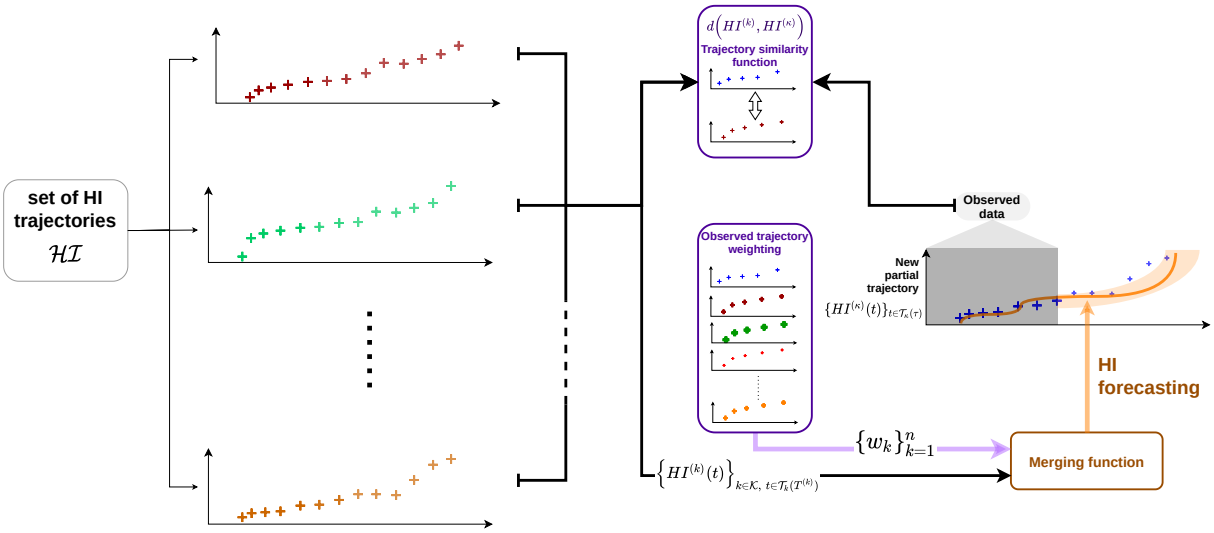


Figure III.9: Schematic view of curve matching HI forecasting.

The main issue with these above introduced approaches is, of course, the need for a high amount of observed run-to-failures trajectories, which is particularly crucial in this setting as the precise observed trajectories are directly used as RUL predictors.

III–2.1.3 Stochastic process model

In this category, are included all the approaches that attempt to model HI trajectories as realisations of a specific stochastic process, hence it includes the stochastic process based approaches described in Subsection II–2.4.1. Herein, both individual and general models settings can be used.

(a) Individual models

For individual models, the goal is to find, for any particular instance κ with a trajectory until $t = \tau$, a stochastic process $X_\omega(t)$ that best explains the observation, i.e. the trajectory. The model chosen to represent the stochastic process can incorporate some physical law, e.g. by defining a trend following this latter. Nevertheless, as opposed to pure physical models, random fluctuations are taken into account and parameterised by the stochastic process model. As a result, X_t is expressed by a deterministic component, e.g. a deterministic trend with parameters Θ and a stochastic component with parameters Ω, e.g. mean and variance of noise or stochastic trend. Overall, the model is denoted $X_\omega(t; \Theta^{(\kappa)}, \Omega^{(\kappa)})$. The parameters $\Theta^{(\kappa)}$ and $\Omega^{(\kappa)}$ are estimated from historical values of the trajectory. For the deterministic parameters $\Theta^{(\kappa)}$, MSE-based optimisation is often used, while for the probabilistic parameters $\Omega^{(\kappa)}$ following probability distributions, maximum likelihood estimation (MLE), is preferred:

$$\tilde{\Omega}^{(\kappa)} = \underset{\Omega}{\operatorname{argmax}}\, \mathcal{L}\left(\{HI^{(\kappa)}(t)\}_{t\in\mathcal{T}_k(\tau)}, \tilde{\Theta}^{(\kappa)}|\Omega\right) \tag{III.3}$$

where $\mathcal{L}\left(\{HI^{(\kappa)}(t)\}_{t\in\mathcal{T}_k(\tau)}, \tilde{\Theta}^{(\kappa)}|\Omega\right)$ is the likelihood of the observed HI for $\tilde{\Theta}^{(\kappa)}$ and given Ω.

Once $X_t(\omega; \Theta^{(\kappa)}, \Omega^{(\kappa)})$ is correctly fitted, the advantage is that it is possible not only to get a single prediction of the HI value for any future instant $t = t+h$, with $h > 0$, but also a full distribution of probabilities for the values of the HI at these instants, which enables a prediction within a confidence interval. It is important to note that this approach often necessitates to simulate the trajectory until it reaches the desired time horizon $t+h$. This diminishes the real capacity to forecast to an infinite time horizon h, as it is obviously not possible to simulate infinitely. In practice, the forecasting step is coupled with an estimation procedure of failure threshold, see Subsection III–2.2, thus giving a stopping criterion for the simulation. A schematic overview of this type of approach is represented in the top of Figure III.10.

Because they base the estimation of parameters only on past values of the current trajectory, individual models suffer from the same limitation as function fitting models when few values have been observed. They therefore rely on the same solution of setting a FPT, see Subsection III–2.1.1. Many possible stochastic process-based models exist, as described in Subsection II–2.4.1, and are particularly adapted for the task at hand. A simple approach proposed in [231] consists of a noisy linear degradation model with a time-varying slope following a white noise process. Gaussian processes have also been investigated, [14] e.g. uses them to model HI trajectories. Therein, the authors define a GP with no trend, that is, with no deterministic parameters, and a mixed covariance function composed by the sum of three kernels: RBF, polynomial and white noise. This work is particularly interesting for its monotonic constraint on the GP and its warping between 0 and 1. Other modelling tools involve mainly Markovian stochastic processes, authors in [93] e.g. propose a linear state-space model (SSM) to predict the HI degradation. the same model was used in [153] and was optimised with N4SID a state-space identification algorithm. The authors of [97] use a non-linear SSM with states following a Paris-Erdogan evolution with a time-varying parameter following a normal distribution. The parameters of their model are calibrated with resorting to a MLE at FPT and then via a particle filter approach for each new measurement available. Lévy processes have also been used in that context. [68] e.g. uses diffusion process with non-linear drift optimised with MLE and [225] follows a similar approach with a non-linear drifted Brownian motion, a MLE based parameters initialisation at FPT, and a unscented particle filter parameters update.

(b) General models

As mentioned at the beginning of Subsection III–2.1, general stochastic processes, now denoted as $X_\omega(t; \Theta, \Omega)$ improve the individual approach, in the sense that they also use trajectories of already observed instances gathered in the set $\mathcal{HI}$ to estimate the parameters of the model. In practice, $\mathcal{HI}$ is used to both set the deterministic parameters Θ and to estimate a prior probability distribution of the stochastic parameters Ω. An empirical distribution of Ω may be obtained from the individual estimation of Ω by MLE for every HI in the set $\mathcal{HI}$. Once a new trajectory is being observed, at any instant $t = \tau$, the distribution of the stochastic parameters Ω might be updated using the Bayes rule:

$$P(\Omega|\{HI^{(\kappa)}(t)\}_{t\in\mathcal{T}_k(\tau)}; \Theta) = \frac{P(\{HI^{(\kappa)}(t)\}_{t\in\mathcal{T}_k(\tau)}; \Theta|\Omega)P(\Omega)}{P(\{HI^{(\kappa)}(t)\}_{t\in\mathcal{T}_k(\tau)}; \Theta)} \propto P(\{HI^{(\kappa)}(t)\}_{t\in\mathcal{T}_k(\tau)}, \Theta|\Omega)P(\Omega) \tag{III.4}$$

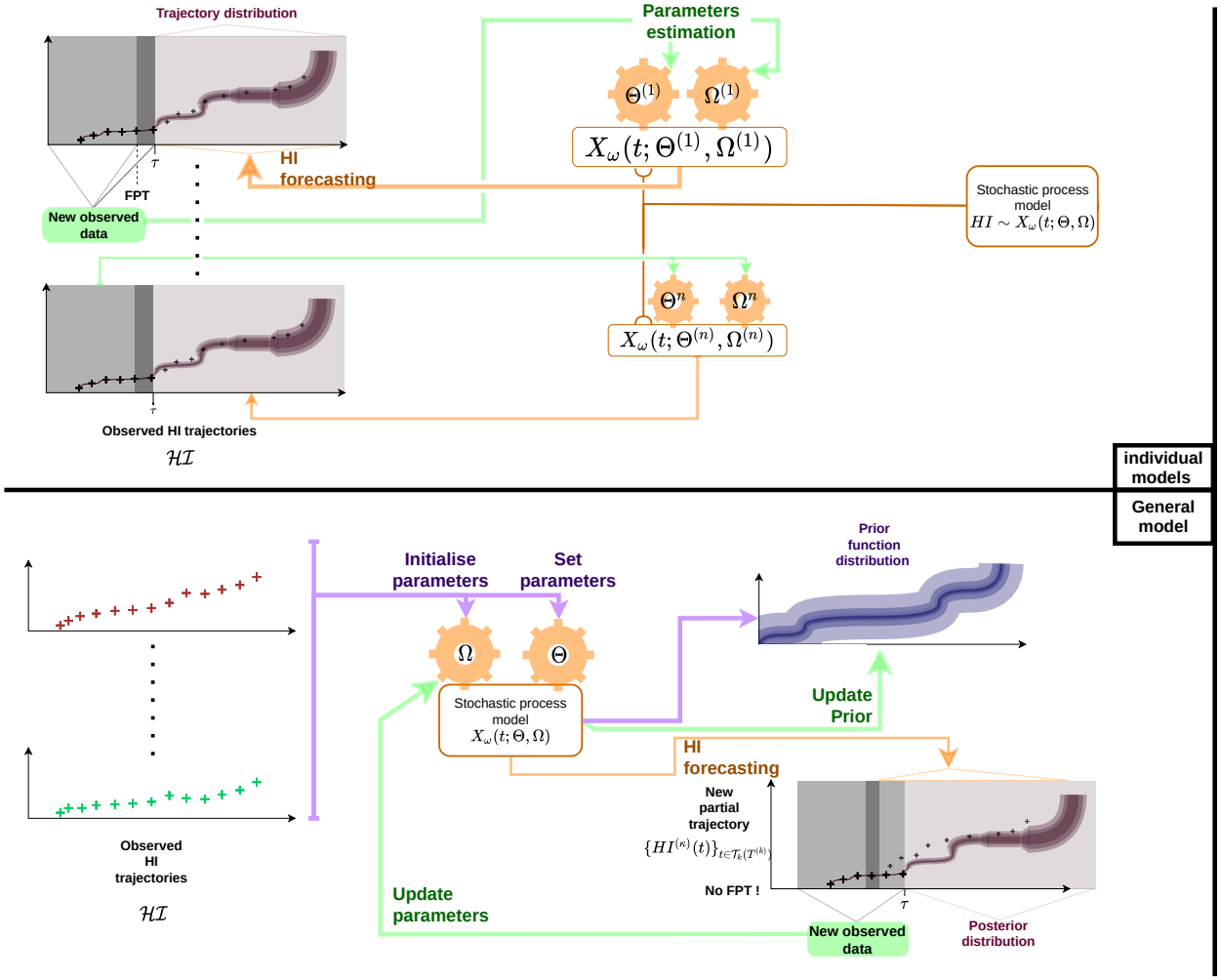


Figure III.10: Schematic views of stochastic process based HI forecasting.

Once the posterior distribution of Ω is updated, it replaces the previous prior $P(\Omega)$ and so on; and enables in turns a prediction on future HI values distribution, similarly as for individual models. A schematic overview of this type of approach is represented at the bottom of Figure III.10. A great advantage of this approach is that it does not need a FPT to start predicting future HI values. It also allows exploiting as much as possible already observed curves, for predicting a new one, an aspect which is completely ignored in individual models setting.

This general model framework has been e.g. used in [54] where the HI is modelled by a exponential function with probabilistic parameters following a multivariate normal distribution. First, each logged observed trajectory is fitted to the logged exponential model, i.e. resulting in a linear model in the log-space. Then, these fitted parameters for each trajectory serve to estimate the distribution of parameters. The parameters update is realised at each new measurement available within a pure Bayesian framework. A general stochastic process model approach is also proposed in [176] where the HI are considered as realisations of a diffusion process with a non-linear drift; the drift being proportional to a "unit-to-unit" parameter following a normal distribution whose parameters are estimated from the MLE based on the observed trajectories. In [101], the authors propose to model the HI with Wiener processes whose drift is also proportional to a "unit-ot-unit" parameter

following a normal distribution, the model parameters are initialised with MLE and the unit-to-unit parameter distribution is updated through time with a Monte-Carlo simulation approach for each new measurement available.

III–2.1.4 Machine learning methods

Machine learning approaches encompass all the ML methods, that are not specialised for modelling stochastic process, but are here customised to perform time series forecasting in a supervised manner. The general formulation of supervised machine learning which can found in II.4 is adapted: The input and output data X and Y are no longer variables of different natures, i.e. explanatory and explained variables. They are here composed of the same variable, or QoI, but at different time instants. X groups the past values of the QoI from the present to a certain past horizon h_p, and Y are the future values of the QoI from the present up to a certain future horizon h_f. Whilst the QoI is the HI, it could be read:

$$\tilde{f}\Big(HI(t-h_p),\dots,HI(t);\Theta_{|\Theta_h},\Theta_h\Big) = HI(t+1),\dots,HI(t+h_f) \quad \forall t \tag{III.5}$$

The exact same models and optimisation procedure of classical machine learning algorithm are used, only the way input and output data are set up is changed. In practice, for setting h_p, a balance must be found between, a large h_p which can improve the model precision by taking into account more past history, and a small h_p that decreases the risk of overfitting if the dataset is too small. Moreover, a larger h_p also induces a further FPT, that is, at least h_p values of the new trajectory must be observed to start the prediction. For h_f, it is in practice impossible to set it such that the time of failure $T^{(k)}$ of any instance k is always included in the future time horizon. Therefore, a small value is often preferred, i.e. $h_f = 1$, and once the model has been learned, the forecasting is performed in a iterative manner, where the forecasted value is used as input in the next iteration, until the failure threshold is reached. Similarly as for the statistical approaches, ML models can be implemented in an individual or a general setting. Nevertheless, there is, in general, no parameter updating when new measurements are available, and ML methods are less suited than stochastic processes for estimating uncertainty. A schematic overview of this type of approach is represented in Figure III.11

Different ML models have been proposed to forecast HI values, of which support vector regression can be cited with e.g. [13] and [122] that respectively use SVR and least-square SVR (LS-SVR) with a RBF kernel. Neural networks are of course also a frequent solution to the issue. Recurrent neural networks (RNN) in particular, which are specialised in processing sequential data have naturally been used for this task, [216] e.g. use LSTM neural networks, [151, 27] rather use gated attention unit (GAU) neural networks, a special implementation of recurrent neural network with attention mechanism incorporated inside the neuron cell, while [233] propose a new consolidated memory GRU neural network. The previously cited examples of approaches are developed in a individual models setting, which is often the case for ML based HI-forecasting. Because they are used for fitting only one trajectory, they require less complex models and faster training time. However, the major drawback is that they must be continuously re-trained after each new HI measurements to keep being accurate, which might be prohibitive for application in real time. Moreover, as mentioned before, they do not take into account observed trajectories ran to failure in the set $\mathcal{HI}$. That is why, general models of HI forecasting can also be developed with ML approaches. [224]

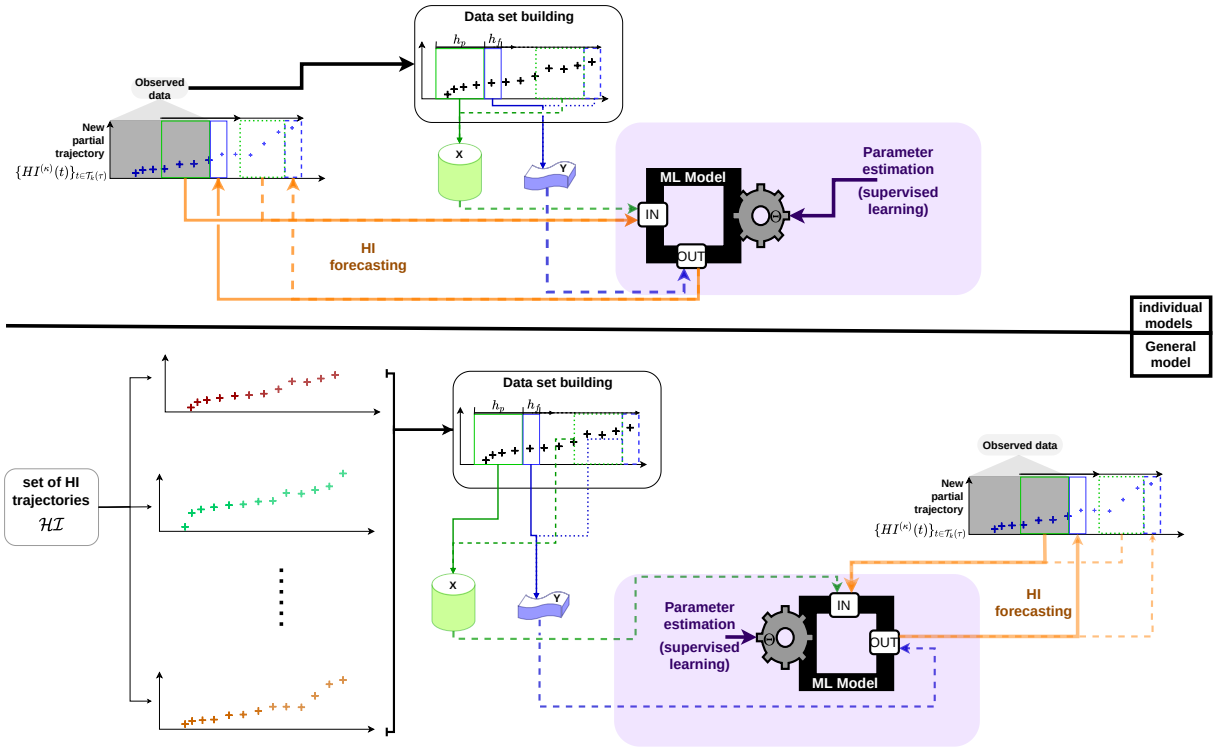


Figure III.11: Schematic view of machine learning based HI forecasting.

e.g. use deep sequence-to-sequence recurrent neural network trained with observed run-to-failures trajectories. If enough of these latter are available, this approach enables to ignore the current trajectory measurements for training and therefore does not need retraining at each measurement. The main drawback here is that the specificities of the new trajectory are not taken into account. This analysis of the existing ML methods brings out the observation that classical supervised ML approaches are not well suited for the task of learning a model based on observed run-to-failures trajectories while also taking into account the new partial trajectory. Indeed, ML approaches are rather used in direct RUL estimation, as mentioned at the beginning of Subsection III–2.1.

As a matter of fact, the issue of taking into account both measurements from observed run-to-failures instances and the new instance has also emerged in the direct RUL setting. An ensemble framework has thus been proposed in [78] that can potentially solve this issue. The authors propose dense NN as weak learners of ensembling approaches, firstly trained separately, with one NN for each observed instances, and then aggregated for forecasting a particular trajectory κ according to their ability to correctly fit this latter. A similar idea developed in Chapter V is partially inspired by this work but also by some specific methods proposed as hybrid approaches considering the current classification of methods.

III–2.1.5 New hybrid approaches

It comes naturally that the proposed classification of approaches here is not perfectly accurate, some particular approaches do not really fit in one precise category but are dispatched in at least two of them. They can be considered in a sense as hybrid approaches. A setting of particular interest is the hybrid approach between stochastic processes and ML methods. The stochastic process modelling setting, especially considered with a general model, is indeed particularly interesting for defining a generic stochastic process whose parameters are first initialised by observed trajectories $\mathcal{HI}$ and then refined at low cost by the new partially observed trajectory. It also enables a statistically grounded way of quantifying uncertainty in prediction. However, these approaches have limitations in defining an adequate function space that could fit any HI trajectories from the set $\mathcal{HI}$, especially if these latter experience diverse modes of failure and thus different degradation dynamics. This is particularly true for the approaches in [176, 101] where the proposed Wiener and diffusion process need to be defined with a non-linear drift function that must take a particular quite restrictive form, here power and exponential functions. ML models and particularly NN are useful for defining wide and complex functional forms. Recently [21] therefore proposes a state space model whose state equation is defined by a dense NN, i.e. a MLP. The authors, as a matter of fact, propose a hybrid approach with state-space stochastic process modelling coupled with a ML model. They use a particular particle filtering approach to recursively update the parameters of an ensemble of MLP which, in effect, simulated a stochastic process model. This proposed approach is restricted to singular models in [21], however, by taking inspiration from [78] as mentioned in Subsection III–2.1.4, an ensemble of MLP can be initialised by all observed trajectories $\mathcal{HI}$ to expand the approach to a general model setting. This particular idea is developed later on in Chapter V and constitutes an original contribution of this thesis.

III–2.2 Threshold estimation

To finally enable RUL estimation, once the forecasting model of the HI is available, the last missing piece is a threshold HI value, that indicates a failed state. In the case of PHI with understandable physical meaning, a threshold can be set by expert knowledge. But in the case of VHI issued from complex systems, no precise physical meaning can be given to any HI value. To estimate a failure threshold, one must therefore rely on already observed HI trajectories and especially their values at the end of life of their respective system's instances. As mentioned in III–1.1 an important property in HI construction is **failure-consistency**, i.e. the fact that HI trajectories from different instances of the same system, reach similar values at the end of their lives. If this property is of paramount importance, it is for this threshold estimation step. Indeed, if all instances reach the same value at the end of their lives, this value can be set as the threshold. Unfortunately, **failure-consistency** is never perfectly complied with, especially when dealing with complex systems with multiple potential mode of failure. Threshold estimation techniques must then be developed to enable more reliable RUL predictions. This threshold estimation can be deterministic, a single value is considered as the threshold. Or it can be probabilistic, the threshold is considered as a random variable that can take a value following a specific probability distribution. Generally speaking, this threshold estimation task seems under-considered in the literature, to the best of the authors' knowledge it is never mentioned in any review on subject of PHM or RUL prognostics [107, 175, 98, 155, 184]. Yet, an attempt of listing the different approaches is given hereafter.

(a) Deterministic constant threshold

Because of its simplicity, the deterministic constant threshold is often the default choice in the existing work. It can be set manually when the HI trajectories have a very high failure consistency, or an understandable physical meaning, as in e.g. [14, 200, 198, 97, 153, 110]. It can also be based on the final values of observed trajectories. For instance, one could derive a threshold as the mean or minimum of the final values of $\mathcal{HI}$. When dealing with complex systems, however, getting HI trajectories that are both monotonic and failure consistent can be very challenging, and when failure consistency is not met, defining a constant deterministic threshold can lead to inaccurate RUL estimations.

These deterministic constant thresholds, however, if simple to implement, are of little help when dealing with cases where failure-consistency is not met. These cases often arise when studying complex systems with multiple failure modes and monitored with highly multivariate time-series that necessitate VHI that have no physical meaning [75, 26, 204]. These cases thus need improved failure threshold estimation methods.

(b) Improved threshold estimation

As mentioned before, the task being under-considered, more sophisticated approach are rare in the literature. Nevertheless, two aspects of the problem have been studied and improved compared to the classical deterministic constant threshold: from deterministic to probabilistic and from constant to adaptive.

A probabilistic threshold can be defined as a probability distribution of the threshold, instead of a single value of the HI above or below which failure is considered. It helps at better capturing uncertainties in the remaining useful life and can lead to more accurate prognostics. Usually, a parameterised probability distribution is assumed for the failure threshold and then the parameters are estimated with MLE based on the last HI values of the observed HI trajectories in $\mathcal{HI}$. For example, [139] use a gamma distribution, [204] a gaussian one and [104] a truncated gamma distribution.

In the case of an adaptive threshold, at each new measurement of the currently observed trajectory, the threshold value is subjected to evolve, be it a deterministic or probabilistic threshold. This setting helps at refining the RUL prognostics at each new observed HI value. This strategy can be observed e.g. in [70] where a deterministic threshold is continuously updated by applying a one-class SVM to determine a hyperplane that separates normal to abnormal values in a hyperspace obtained with kernel functions. This hyperplane is considered as the failure threshold, and the OC-SVM is continuously retrained as more HI value are observed. In [26, 117, 208] the authors also resort to an adaptive threshold but with a different approach. They consider a unique parameterised degradation model for all observed instances and fit this latter for each available HI trajectories with MLE. When a new instance HI trajectory is partially observed, the model is also fitted on it. Additionally, this partial trajectory is approximated by a weighted average of all the observed trajectories, where the weights are optimised with MSE minimisation. The obtained weights can be considered as similarity values between the partial trajectory and each observed one. These latter, are finally

used to estimate a failure threshold as the weighted average of historical HI trajectories last values. This strategy was originally proposed in [26] and then enhanced in [208] that turned the threshold as a probabilistic one following a Gaussian distribution with the mean equal to the previous deterministic threshold and a variance estimated by the empirically on the historical HI trajectories.

The idea of an adaptive probabilistic threshold seems like the best option for one who wishes to better quantify uncertainty and refine a RUL estimation as more HI values are observed. The methodology proposed in [26, 117, 208] is thus used in this work and is adapted to the proposed RUL estimation approach in Chapter V. More particularly, it is adapted to a Monte-Carlo-based HI forecasting with particle filter, which is a common strategy in the literature. Our proposed adaptation of the probabilistic adaptive failure threshold can thus be reused in numerous HI forecasting settings using particle filters and is therefore an important contribution of this thesis.

IV

Health indicator construction: proposed approach

As mentioned in Subsection III–1.2.2, two main types of approach exist for constructing a HI model of a physical system whose multiple parameters are monitored to collect MTS. It was also found that few of them enforce monotonicity constraints on the constructed HIs.
The **fusion-based strategy** has nevertheless benefited from some recent developments in that direction, in particular with GA approaches that enabled to incorporate some monotonicity constraints during the optimisation or learning procedure, more precisely as a constraint in the objective function of the HI model [208, 138]. Even more recent developments, with deep learning tools, have also been proposed in the fusion-based setting for incorporating monotonicity constraint inside the loss of the NN [150]. However, the second main setting of HI model construction, a.k.a. **similarity-based strategy** has not yet, to the best author's knowledge, benefited from the development of recent proposition incorporating monotonicity constraint during the optimisation or learning.
It is important to point out that this **similarity-based strategy**, when adapted to deep learning, which does not need to prove its performance in extracting complex patterns in data any more, does enable the introduction of monotonicity constraint. This idea is the core contribution of this chapter, as one solution to overcome the existing limitations of recently proposed methods raised in the bibliography made in Chapter III.
More specifically, in this chapter, we propose a similarity-based HI construction approach, built upon a contrastive learning setting, by means of a siamese neural network (SNN) as the representation learning model. An adapted training samples selection and a specific constraint incorporated directly in the contrastive loss function are proposed to enforces monotonicity of the resulting HI. No additional hypothesis on the evolution of the HI is therefore needed, which is a clear improvement from the literature and makes the proposed method very generic and adaptable to any physical system. The proposed method neither focuses on a particular and fixed deep learning architecture nor on a specific signal preprocessing technique. It is rather based on a generic methodology using similarity learning, SNN and contrastive loss applied to the problem of HI construction, which can be easily adapted to any input data and application context. Firstly, an overview of the concepts and methods from deep similarity learning and contrastive learning, used in the present work, is given in Section IV–1. The enhancement of these methods to fit the HI construction problem is then presented in Section IV–2. Next, the application of the proposed methodology to two datasets used for RUL prediction is presented in Subsection IV–3.1 and Subsection IV–3.2 with a detailed assessment of the performance obtained. Additionally, because it also proposes an approach based on deep self-supervised learning with a monotonicity constraint, the work of [150] is thoroughly compared to the proposed method.

IV–1 Similarity learning

This section recalls the main concepts of similarity learning and focuses on its deep learning version: the SNN and contrastive triplet loss.

IV–1.1 Similarity learning

Similarity learning or metric learning is a field of machine learning that aims to infer a distance function between samples of a given dataset directly from this latter [92]. In mathematics, a distance function, a.k.a. a metric, $d : D \times D \to \mathbb{R}$ where D is a given set, must satisfy the four properties defined by eqs. (IV.1) to (IV.4) for any pair of elements $(a, b) \in D$ [59].

$$\text{Non-negativity: } d(a,b) \geq 0 \tag{IV.1}$$

$$\text{Identity of discernable: } d(a,b) = 0 \Leftrightarrow a = b \tag{IV.2}$$

$$\text{Symmetry: } d(a,b) = d(b,a) \tag{IV.3}$$

$$\text{Triangle inequality: } d(a,c) \leq d(a,b) + d(b,c) \tag{IV.4}$$

It is worth noting that a function meeting all the aforementioned criteria except for Eq. (IV.2) is labelled as a pseudo-distance. Unlike the concept of classical and analytical distance metrics, such as Euclidean or Manhattan distances, which rely on defined mathematical expressions of d, the principle of **similarity learning** is to infer the function d as $d_\theta(a,b) = f(a,b;\theta)$ where θ is a set of parameters estimated from a given data set $\{x^{(j)}\}_{j=1}^{n}$ where $x^{(j)} \in D$ such that d_θ satisfies the properties of eqs. (IV.1) to (IV.4) or alternatively of eqs. (IV.1), (IV.3) and (IV.4) for a pseudo-distance. Typically, such a learned metric is sought to better compare the elements of D with respect to a specific task, e.g. clustering, ranking, matching predefined similarity judgments. A simple yet illustrative example of learned distance metrics is the Mahalanobis pseudo-distance [163] given by Eq. (IV.5):

$$d_{maha}(a,b) = \sqrt{(a-b)^T M (a-b)} \tag{IV.5}$$

where a and b are assumed to be identically distributed from a joint probability distribution of covariance matrix M. The distribution being unknown in practice, the covariance matrix M is therefore estimated from a given dataset issued from the same distribution. The estimation $\widehat{M}$ of M therefore stands for the θ parameters of the d_θ metric as defined above.
Similarity learning has proven to be effective for clustering high-dimensional objects such as images [170, 215, 82, 232], but also for recommendation systems [112, 81], ranking objects [22] or face verification [170]. The recent breakthrough of deep learning has opened several opportunities in the area of similarity learning. Among all, self-supervised contrastive learning approaches based on SNN [19, 31, 170] have emerged and offer an appealing setting for HI learning.

IV–1.2 Siamese neural networks

As discussed in Subsection II–2.5, Neural Networks (NN) are machine learning models whose mathematical architectures, which depend on the learning task to be performed, generically consist in stacking and interconnecting multiple layers of neurons. Each neuron linearly combines its inputs

and parameters, the result of which is passed through a so-called activation function which gives the final output of the neuron, then forwarded as input to the neurons of the next layer. Parameters of NN, i.e. weights and biases, are typically optimised using stochastic gradient descent to minimise a loss function when fed with example data. Unlike a classical NN, an SNN is composed of two or more identical NN, i.e. clones of a given neural architecture, which all have the same parameters, see Figure IV.1. It means that whilst training a SNN, its clone NNs, i.e. subnetworks, are updated simultaneously. The output layer of each subnetwork generates comparable vectors in an **embedding** or **latent space** of size e, where e is the number of neurons or outputs in the output layer. In this latent space, a comparison operation of two or more input samples is performed either by a classical distance, such as the Euclidean distance, or by a neural distance layer. Figure IV.1 shows the standard architecture of a SNN with two identical subnetworks. By construction, a SNN can be used for dimensionality reduction [230] or similarity learning [19, 31]. The particularity of the SNN architecture and the learning task it must perform requires a particular form of loss function, called contrastive loss, which is described in the next section.
It is due noting that using a valid metric or pseudo metric in the comparison layer ensures that the learned metric d_θ is a pseudo-metric. Indeed, the identity of discernable defined in Eq. (IV.2) cannot be ensured, as the NN subnetworks are not necessarily bijective functions.
An important point to bear in mind, is that it is the shared parameters of the SNN's subnetworks that completely condition the ability of the distance function used in the latent space to solve the task at hand, which is to discriminate samples.

IV–1.3 Contrastive triplet loss

The idea of contrastive loss as well as SNN comes with the dimensionality reduction task. A projection from high to low dimensional spaces is considered valuable when it maps inputs considered dissimilar to distinct enough vectors in the output space, and inputs considered similar to closely related vectors. A loss function should thus be designed such that, a SNN minimising it produces a mapping function that reaches the aforementioned objective. A general formulation of the contrastive loss has been given by [65] for a SNN composed of two clones, therefore fed by sample pairs. It can be referred as duet contrastive loss and reads:

$$\mathbb{L}(\theta) = \sum_{j=1}^{N_p} L(\theta, Y^{(j)}, (X_1, X_2)^{(j)}) \tag{IV.6}$$

$$\text{where: } L(\theta, Y^{(j)}, (X_1, X_2)^{(j)}) = \sum_{j=1}^{N_p} (1 - Y^{(j)}) L_S(d_\theta^{(j)}) + Y^{(j)} L_D(d_\theta^{(j)}) \tag{IV.7}$$

$$\text{and: } d_\theta^{(j)} = d_\theta(X_1^{(j)}, X_2^{(j)}) \tag{IV.8}$$

where N_p is the number of training pairs, θ represents the parameters of the SNN, $(X_1, X_2)^{(j)}$ stands for an input sample pair, $d_\theta^{(j)}$ the in-between Euclidean distance of the latter's projection in latent space, and $Y^{(j)}$ corresponds to the binary label indicating if the pair is similar ($Y^{(j)} = 0$) or dissimilar ($Y^{(j)} = 1$). The partial loss functions L_S and L_D, respectively for a couple of similar and dissimilar inputs, are expressed in eqs. (IV.9) and (IV.10) as proposed by the authors of [65]. They are functions of $d_\theta^{(j)}$ and are respectively activated for a pair of similar ($Y^{(j)} = 0$) and dissimilar

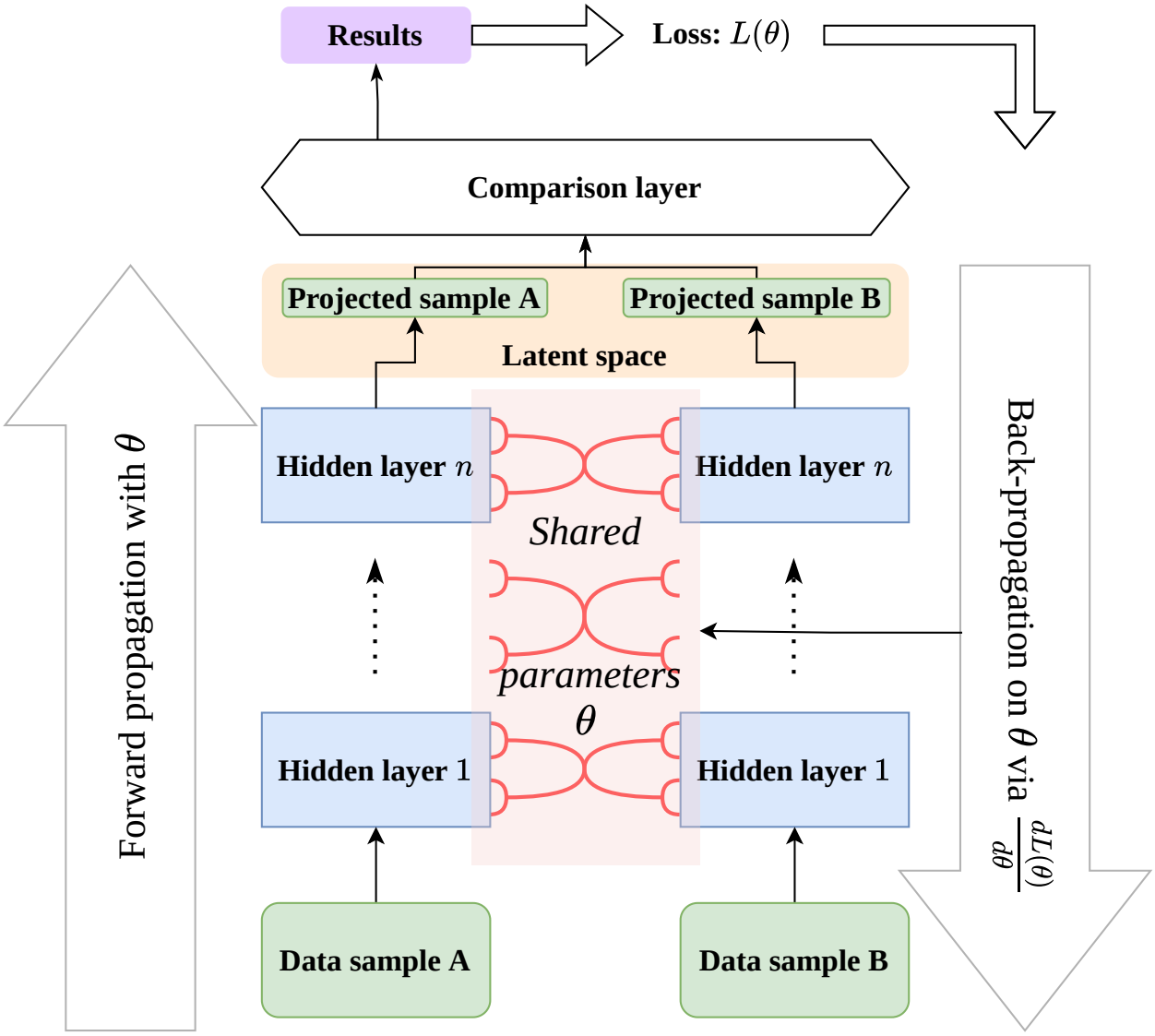


Figure IV.1: Generic representation of a SNN.

samples ($Y^{(j)} = 1$).

$$L_S(d_\theta^{(j)}) = \frac{1}{2}(d_\theta^{(j)})^2 \tag{IV.9}$$

$$L_D(d_\theta^{(j)}) = \frac{1}{2}\max(0, m - d_\theta^{(j)})^2 \tag{IV.10}$$

where $m > 0$ is a margin parameter. Here, when a pair of similar samples is passed through the SNN, it is penalised by L_S for a high in-between distance of the pair. Inversely for a dissimilar pair the SNN is penalised by L_D for an in-between distance lower than m.

In the contexts of face reidentification and image clustering, recent works [170, 201] have proposed a new version for the contrastive loss, adapted to triplets rather than pairs of samples. This triplet loss reads:

$$L(\theta, (A, P, N)^{(j)}) = \max(0, d_\theta(A^{(j)}, P^{(j)}) - d_\theta(A^{(j)}, N^{(j)}) + m) \tag{IV.11}$$

where $(A, P, N)^{(j)}$ is a triplet of samples, A denoting the anchor (a reference sample to be compared to), P a positive sample (expected close or similar to the anchor), and N a negative sample (expected dissimilar or distant from the anchor). The role of m in eqs. (IV.10) and (IV.11) is similar and crucial for the training of SNN. It enforces the SNN to push the negative samples far from the anchor sample in the latent space, while also relaxing this constraint once the negative is sufficiently far. When $d_\theta(A^{(j)}, N^{(j)}) - d_\theta(A^{(j)}, P^{(j)})$ in Eq. (IV.11) or $d_\theta^{(j)}$ in Eq. (IV.10) becomes greater than m for a given triplet $(A, P, N)^{(j)}$, then L is set to zero, hence the model does not update its weights. If the loss reaches zero for every possible triplet, then the positive and negative samples would be perfectly separable in the SNN latent space. In practice, a zero loss is seldom reached because of the complexity of real data, but driving the loss close to zero makes the positive and negative samples more separable, improving their clusterability. An overview of SNNs trained with triplet loss for image similarity learning is represented in Figure IV.2. Furthermore, if one is able to estimate the extent to which $P^{(j)}$ and $N^{(j)}$ should be distant, the margin m in Eq. (IV.11) can be defined as follows for each triplet $(A, P, N)^{(j)}$:

$$m^{(j)} = f(r, (A, P, N)^{(j)}) \tag{IV.12}$$

where r is a relevance score function which relates to the similarity of two samples in the application context, and f is the margin function that computes the margin $m^{(j)}$ for a specific triplet and the relevance function r. This leads to a variant of the original triplet loss, called adaptive triplet loss introduced in [64], where the following version of f was proposed:

$$f(r, (A, P, N)) = \left| \frac{r(A, P) - r(A, N)}{\max\limits_{X,Y} r(X, Y)} \right| \tag{IV.13}$$

The function r is context-dependent and is defined later in this chapter in Subsection III–1.2.2 when applying contrastive loss to the context of HI learning. With contrastive triplet loss, [170, 201] have achieved state-of-the-art results on face reidentification and image similarity learning for classification or clustering. Triplet loss has since raised interest for training SNN and has been used in many applications, e.g. dimensionality reduction [185], knowledge distillation or NN compressing [140], fMRI-based brain disorder classification [76], enzyme function classification [125], video-based emotion recognition[95] or intention detection in spoken languages [158]. Yet, to the best of the authors' knowledge, no work has been done on using SNNs trained with triplet loss in order to learn a distance metric for similarity-based HI construction. This chapter therefore focuses on filling in this gap in the literature, hoping to provide a new path for further works in this direction. The next section develops this idea and describes the proposed loss functions as well as the preparation of the (A, P, N)-samples, referred to as *triplet selection*. This point is of paramount importance for training SNN with triplet loss and is guided by the user on the application context.

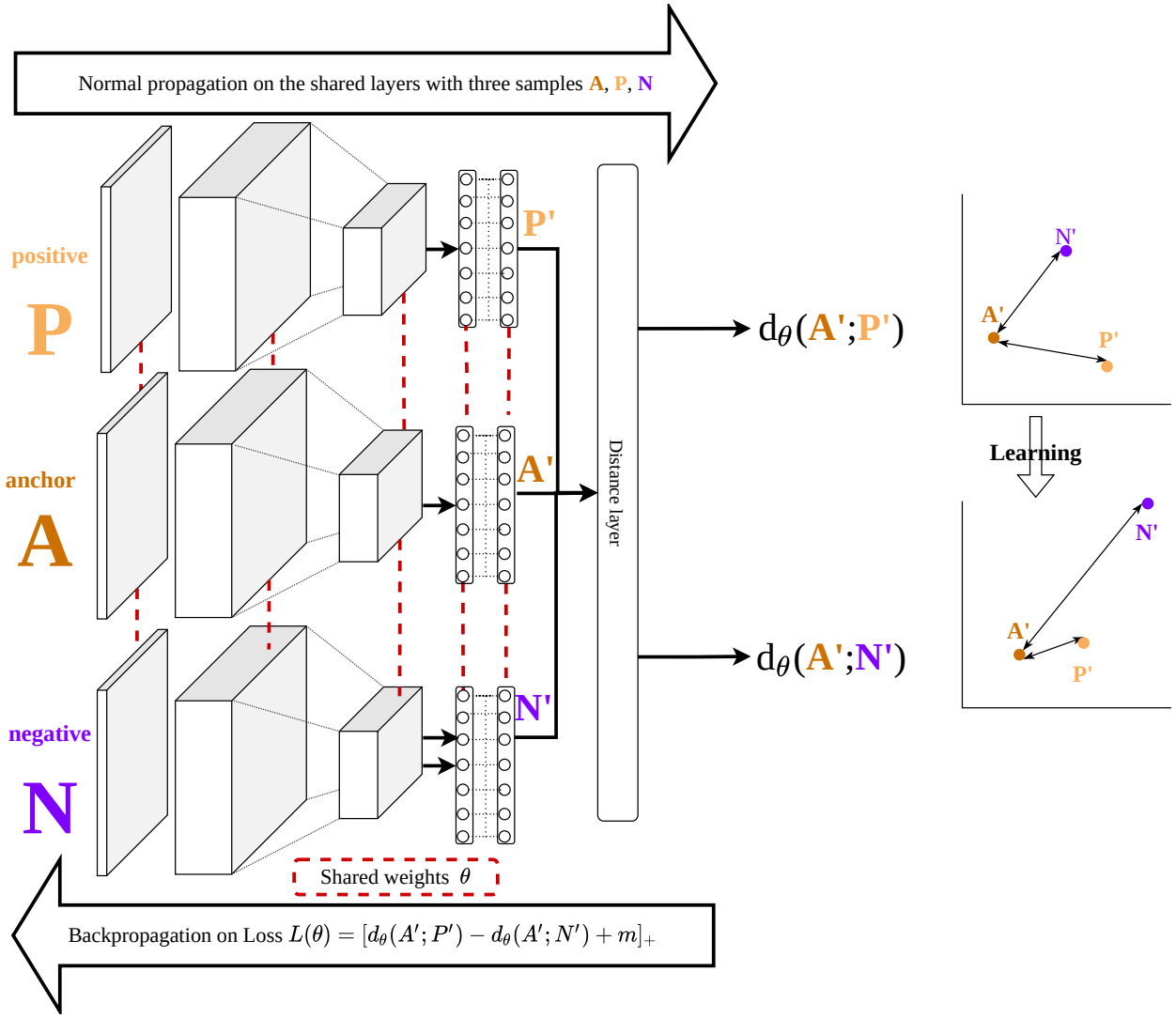


Figure IV.2: Original triplet loss with image inputs.

IV–2 SNN triplet loss based HI

This chapter proposes novel similarity-based methods for HI construction whose distance metric d_θ is learned with a SNN and triplet loss. The proposed HI should further ease the RUL estimation and therefore meet some requirements which are recalled hereafter.

IV–2.1 Health indicator construction

Let us consider that sensors record several variables of multiple instances from a given system or device throughout their lifetimes. We assume that the collected MTS can capture the underlying physical degradation phenomena of the monitored system. This section outlines our contribution to the health assessment task with a novel methodology for processing these MTS with the objective of producing a HI that gives the health state of the system at any time. It relies on learning a distance metric d_θ between subseries of a MTS using contrastive loss based SNN. It must be

recalled that, among other considerations, the constructed HI must increase monotonically with time from a perfectly healthy state at the beginning of the system's use to a degraded or failed state at the end of its life. Three strategies for enforcing the monotonic constraint during model training have been studied and will be presented below. An overview of the proposed approach is given in Figure IV.3 and relies on three major steps:

- preprocessing of data and time windowing,
- definition of the SNN architecture,
- definition of the loss function of the SNN and selection of training samples.

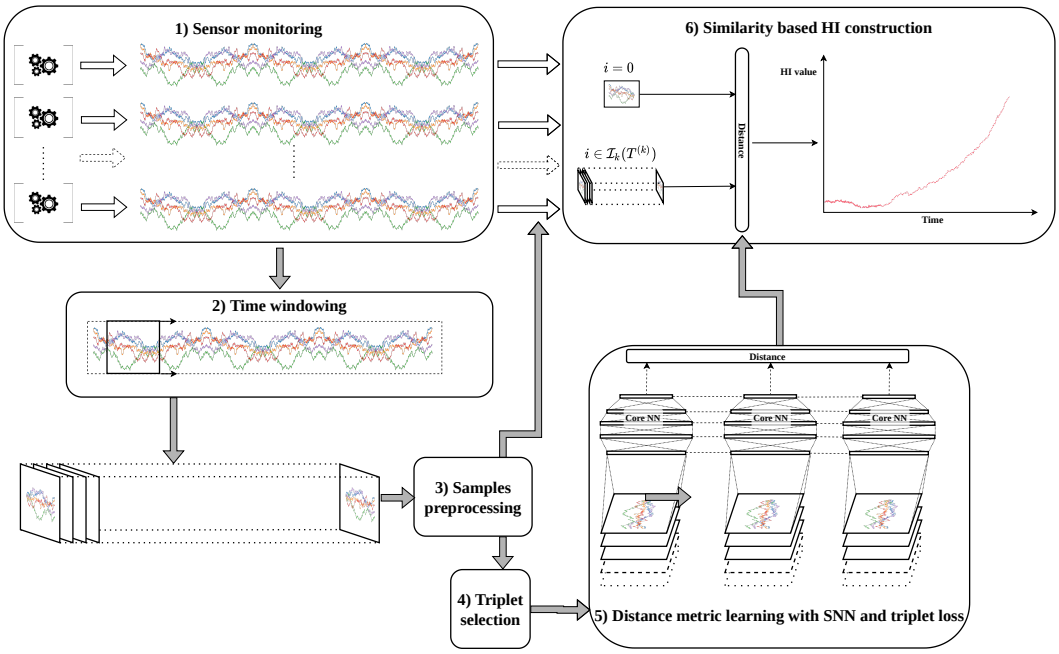


Figure IV.3: Overview of HI construction based on SNN and triplet loss.

IV–2.1.1 Preprocessing and time windowing

The raw MTS of the n instances of the studied system are denoted $\{s^{(k)}(t)\}_{k\in\mathcal{K},\, t\in\mathcal{T}_k(T^{(k)})}$ where $s^{(k)}(t) \in \mathbb{R}^d$ are the values recorded for the d signals at time t and where $T^{(k)}$ is the time of failure of the k^{th} instance of the studied system. These MTS should first be preprocessed. This preliminary and optional step aims at denoising the data and extracting useful features, in order to further ease the similarity learning. One should refer to references listed in Table III.2 for more details on possible techniques. Additionally, MTS, if uniformly sampled, also must be windowed during the preprocessing to be transformed in MTS with subseries-uniform sampling, which is a prerequisite of any NN-based method. Raw or preprocessed MTS $\{s^{(k)}(t)\}_{k\in\mathcal{K},\, t\in\mathcal{T}_k(T^{(k)})}$ are therefore split into subseries $\{s^{(k)(i)}(t)\}_{k\in\mathcal{K},\, i\in\mathcal{I}_k(T^{(k)}),\, t\in\mathcal{T}_{ki}}$ of equal length where $\mathcal{I}_k(T^{(k)}) = \{0, \ldots, I^{(k)}\}$, $I^{(k)} = \frac{T^{(k)}-\Delta_l}{\Delta_G}$, and where $s^{(k)(i)} = \{s^{(k)}(t = i\cdot\Delta_G), \ldots, s^{(k)}(t = i\cdot\Delta_G + \Delta_l)\}$ is the i^{th} subseries of the kth instance of the studied system, Δ_l is the window length and Δ_G is the sliding step of the sliding

window operation. This process is illustrated in Figure IV.4. $s^{(k)(i)}$ are assumed to be stationary, in contrast to the whole time series from which they are extracted. The values of Δ_G and Δ_l depend on the application context and are therefore discussed in Section IV–3.

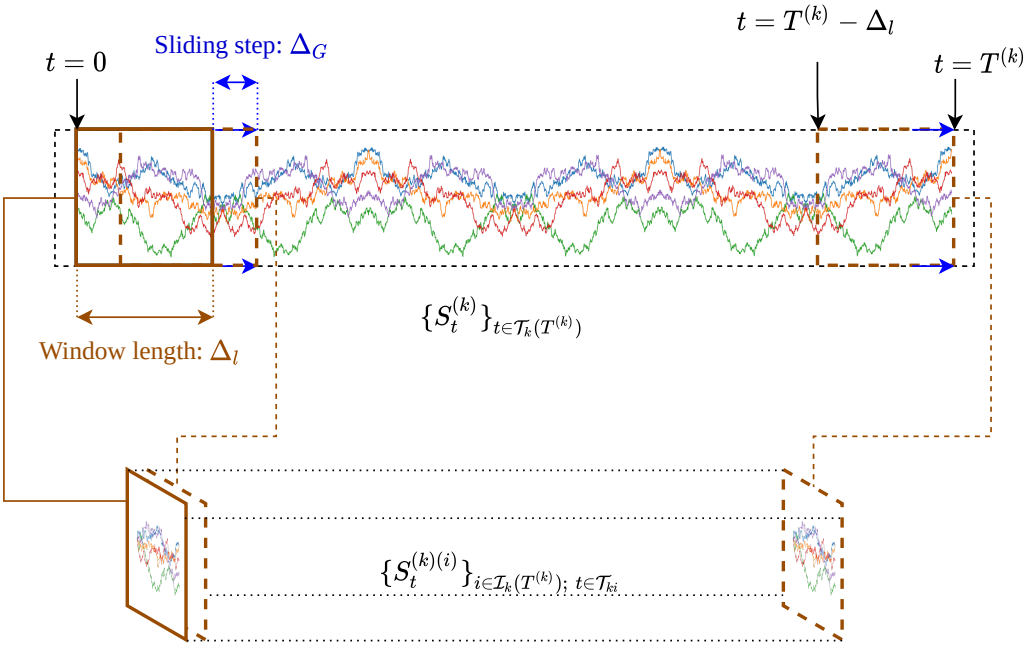


Figure IV.4: The sliding window operation.

IV–2.1.2 Definition of the SNN architecture and HI definition

The choice of architecture for the clone networks NN_θ that compose the SNN depends on the preprocessing step of the multivariate time series and is therefore application-dependent. This point is further discussed in Section IV–3.
As far as the objective of the proposed SNN is to learn a distance function d_θ between any two subseries $s^{(k)(i)}$ and $s^{(k)(j)}$, the HI we proposed merely reads:

$$\begin{aligned} HI(t_i)^{(k)} = HI_i^{(k)} &= d_\theta(s^{(k)(0)}, s^{(k)(i)}) \\ &= d\left(e_\theta^{(k)(0)}, e_\theta^{(k)(i)}\right) \end{aligned} \tag{IV.14}$$

where $t_i = i \cdot \Delta_G + \Delta_l$, d denotes for the metric used in the distance layer in Figure IV.2, $e_\theta^{(k)(0)}$ and $e_\theta^{(k)(i)}$ denotes for the representations of the two subseries $s^{(k)(0)}$ and $s^{(k)(i)}$ in the latent space. Under the hypothesis that the subseries $\{s^{(k)(i)}(t)\}_{k \in \mathcal{K},\, i \in \mathcal{I}_k(T^{(k)}),\, t \in \mathcal{T}_{ki}}$ contain information on the degradation state of the system, the aim is to train the SNN such that the representation of these subseries in its latent space reflect this degradation state. More precisely, the aim is that the distance d_θ between two subseries of the same system at two significantly different instants reflect

the difference of degradation of the system at these two instants, thus the subseries representations at these two instants must have significantly different statistical contents. Therefore, for the healthy state, at $i = 0$, $HI(k)(0) = 0$ is desired, whereas at $i = I^{(k)}$, when the system can no longer operate in standard conditions, $HI^{(k)}(I^{(k)})$ is expected to be the highest value for any i.

IV–2.1.3 Definition of the loss function of the SNN and selection of training samples

This step is the major contribution of this chapter, and three different strategies are proposed to reach the objective, i.e. learn a distance d_θ that enables to represent the health status of the system as per Eq. (IV.14).

(a) Naive strategy

This first naive approach, described precisely in Algorithm 1, consists in applying a classical triplet loss, as defined in Eq. (IV.11), and a basic selection of each triplet samples (A, P, N) of the training set as follows: For each **epoch** and for every sample in $\{s^{(k)(i)}\}$ considered as an anchor A, a positive sample $P = s^{(k)(p)}$ is randomly chosen in the neighbourhood of A defined by the hyperparameter η to be fine-tuned. The negative sample is chosen away of the neighbourhood of A, see Figure IV.5 for a visual illustration. We here briefly remind that an epoch correspond to all the training steps necessary for a NN to have seen all the training samples. A NN is commonly trained over several dozens of epochs. Here an epoch correspond to a training cycle of the SNN where every sub-series that possibly can is considered once as an anchor.

This approach is the strategy that is the closest to the original idea of triplet loss, where only the similarity of samples in the same neighbourhood is ensured. However, it is not necessarily expected that this approach leads to a monotonous HI. Indeed, the possible issue with this naive strategy is that, the ordering of (A, P, N) in time is not enforced by the classical triplet loss. It only ensures that $d_\theta(A, P) < d_\theta(A, N)$, but N and P can both be selected in the past or future of the anchor. The model has therefore no incentives to map the samples to a latent space where A, P and N are somehow placed relatively to their monitoring time.

(b) Anchor-at-start strategy

A second strategy, described by Algorithm 2, is here proposed. This strategy guides more precisely the selection of (A, P, N) triplets during the model training to meet the monotonicity constraint. The loss function is the same as in Eq. (IV.11). In contrast with the *naive* strategy where every sample is in turn considered as an anchor A, here every sample $s^{(k)(i)}$ is in turn considered as a negative N. The anchor A is always randomly chosen in the past of $N = s^{(k)(i)}$, in the neighbourhood of the system life beginning. The positive P is then chosen between A and N in a so-called, "*semi-hard*" neighbourhood defined by both hyperparameters η and λ, see Figure IV.6 for a visual representation. $\lambda > 0$ is a crucial hyperparameter to be fine-tuned, it defines the minimal time distance $\lambda\eta$ a positive P must have with the negative N. If $\lambda = 0$ then P is in the neighbourhood of N and the separation of the two is hard. If λ is too high, then P is very far from N and the separation

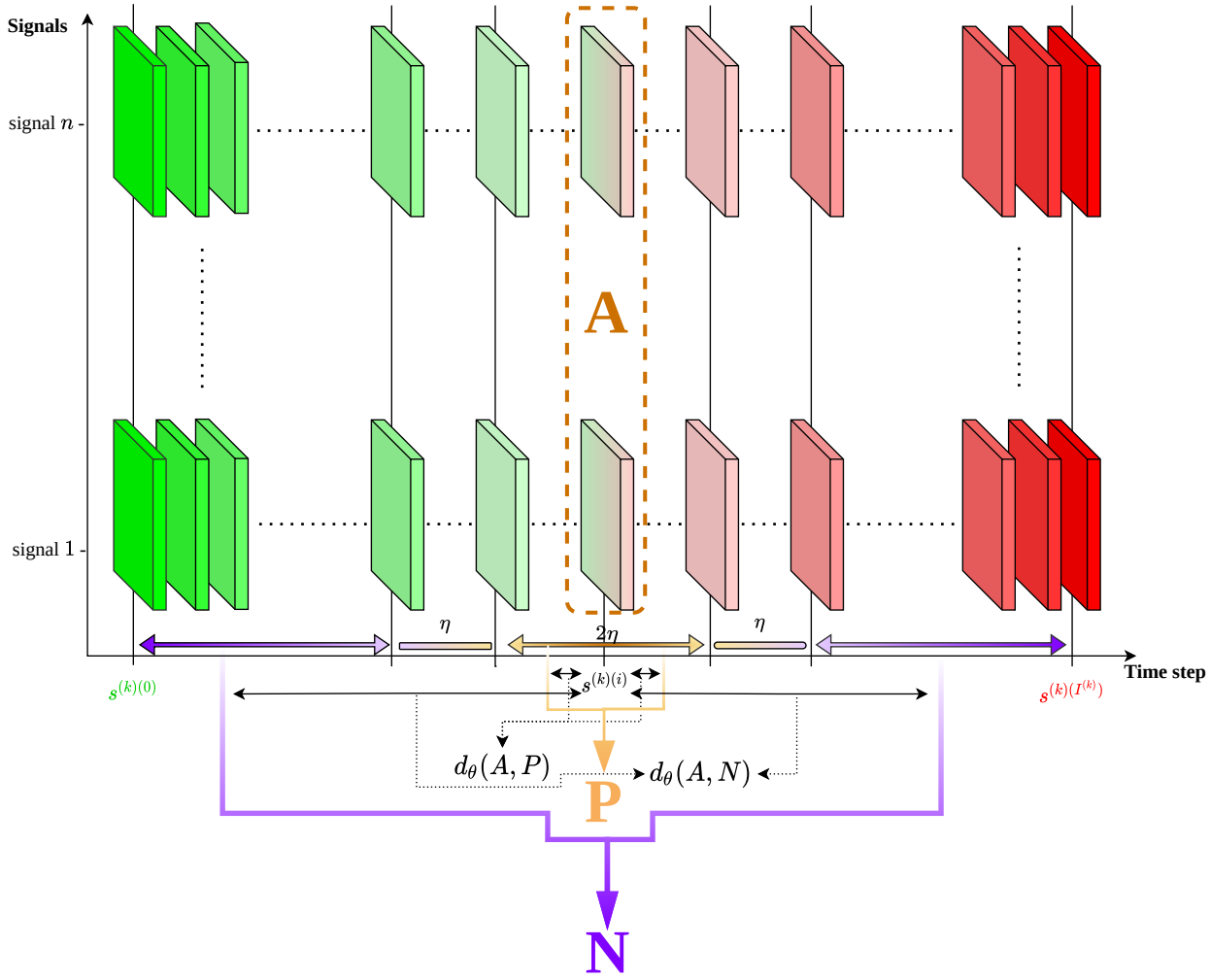


Figure IV.5: Naive strategy for adapting triplet loss to HI construction.

is too simple, thus the distance metric is not precise. λ must then be carefully tuned to correctly define the *"semi-hard"* margin with which the learning of an accurate distance metric can be performed.

With this approach, a ranking constraint is also enforced, with $d_\theta(A,P) < d_\theta(A,N)$ which correspond to the order of monitoring times $t_A < t_P < t_N$. This approach does not necessitate the introduction of another sample to form a quadruplet instead of a triplet. Additionally, this approach focuses only on comparing samples at any monitoring time with their respective ancestors at the monitoring start, which is what the SNN model is used for in the end for the HI construction.

(c) Double-negative strategy

Finally a third approach, described by Algorithm 3 relies on the construction of a quadruplet SNN inputted with four samples (A, P, N_p, N_f) selected among the subseries $\{s^{(k)(i)}\}$ rather than a triplet. For every sample in $\{s^{(k)(i)}\}$ considered as an anchor A, a positive sample $P = s^{(k)(p)}$ is randomly chosen in the neighbourhood of A defined by the hyperparameter, η similarly as the previous approach. Then, two negatives subseries $N_p = s^{(k)(n_p)}$ and $N_f = s^{(k)(n_f)}$ are randomly chosen away from A's neighbourhood in its past and future. Besides the initial goal of triplet loss

Algorithm 1 Naive triplet selection for **one epoch**.

n is the number of signals $s^{(k)}$
S is the list of list of sub-signals $s^{(k)(i)}$ ▷ $S[k][i] = s^{(k)(i)}$
η is an integer and the neighbourhood hyperparameter
Triplets is an empty list of quadruplets
for $k \in 1 : n$ **do**
  $s^{(k)} \leftarrow S[k][:]$
  $I^{(k)} \leftarrow \text{len}(s)$
  for $i \in 1 + 2\eta : I^{(k)} - 2\eta$ **do**
    $a \leftarrow s^{(k)}[i]$
    $ind_p \leftarrow \text{random\_uniform\_pick}([i - \eta, i - 1] \bigcup [i + 1, i + \eta])$
    $ind_n \leftarrow \text{random\_uniform\_pick}([1, i - 2\eta] \bigcup [i + 2\eta, I^{(k)}])$
    $p \leftarrow s^{(k)}[ind_p]$
    $n \leftarrow s^{(k)}[ind_n]$
    $Triplets.\text{insert}((a, p, n))$
  end for
end for

Algorithm 2 Anchor-at-start triplet selection for **one epoch**.

n is the number of signals $s^{(k)}$
S is the list of list of sub-signals $s^{(k)(i)}$ ▷ $S[k][i] = s^{(k)(i)}$
η is an integer and the neighbourhood hyperparameter
λ is an integer and the semi-hard margin hyperparameter
Triplets is an empty list of triplets
for $k \in 1 : n$ **do**
  $s^{(k)} \leftarrow S[k][:]$
  $I^{(k)} \leftarrow \text{len}(s)$
  for $i \in 1 + (2 + \lambda)\eta : I^{(k)}$ **do**
    $n \leftarrow s[i]$
    $ind_a \leftarrow \text{random\_uniform\_pick}([1, 1 + \eta])$
    $ind_p \leftarrow \text{random\_uniform\_pick}([i - (\lambda + 1)\eta, i - \lambda\eta])$
    $a \leftarrow s^{(k)}[ind_a]$
    $p \leftarrow s^{(k)}[ind_p]$
    $Triplets.\text{insert}((a, p, n))$
  end for
end for

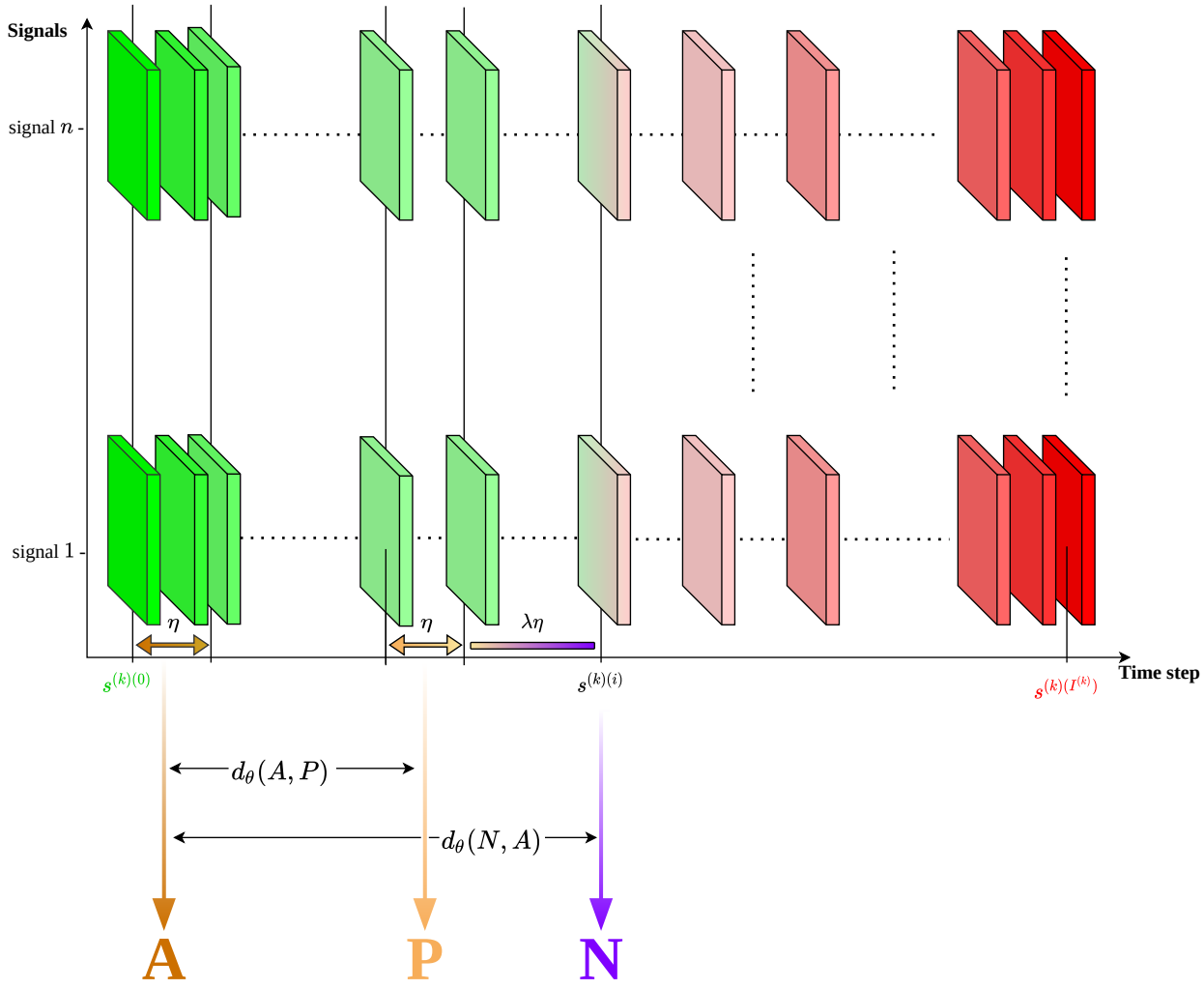


Figure IV.6: Anchor at start strategy for adapting triplet loss to HI construction.

of separating anchor and positive samples from negative ones in the SNN latent space, the idea is to force the learned metric distance d to respect the ordering of N_p, A and N_f in time, that is $d(N_p, A) < d(N_p, N_f)$, see Figure IV.7. To do so, the original loss function in Eq. (IV.11) is modified as follows:

$$\begin{aligned} L(\theta, (A, P, N_p, N_f)^{(j)}) = &\max(0, d_\theta(A^{(j)}, P^{(j)}) - d_\theta(A^{(j)}, N_p^{(j)}) + m) \\ &+ L_{\text{pen}}(\theta, (A, N_p, N_f)^{(j)}) \end{aligned} \tag{IV.15}$$

$$L_{\text{pen}}(\theta, (A, N_p, N_f)^{(j)}) = \max(0, d_\theta(A^{(j)}, N_p^{(j)}) - d_\theta(N_p^{(j)}, N_f^{(j)}) + m_{\text{pen}}) \tag{IV.16}$$

The first term in Eq. (IV.15) is the same as in Eq. (IV.11) with N_p instead of N, responsible for the separation of positive and negative samples. The second term, denoted L_{pen} defined in Eq. (IV.16) and embodying the same mathematical form as in Eq. (IV.11), is a penalisation term, in turn responsible for the correct ordering of N_p, A and N_f. Indeed, N_p is first forced to be projected far from A compared to P with the classical loss, and then simultaneously close to A compared to N_f with the additional penalisation. This forces the SNN to always project N_f and N_p as the most further away samples and A somewhere in between. This way, it is expected to obtain monotonous

$HI(t)$ curves. A new margin parameter m_{pen} is introduced in Eq. (IV.16), an hyperparameter to be fine-tuned that acts similarly in L_{pen} as m does in L.

In this approach, the combination of loss and selection strategy leads to a ranking constraint with $d_\theta(A,P) < d_\theta(A,N_p) < d_\theta(N_p,N_f)$ which corresponds to the order of monitoring times $t_{N_p} < t_A < t_{N_f}$ and $t_{N_p} < t_P < t_{N_f}$. Only the ordering of A and P is undefined, which is not a major issue since both are constrained to be represented close to each other. The only drawback, however, is the introduction of a supplementary negative sample, which leads to a quadruplet SNN that is more computation-demanding.

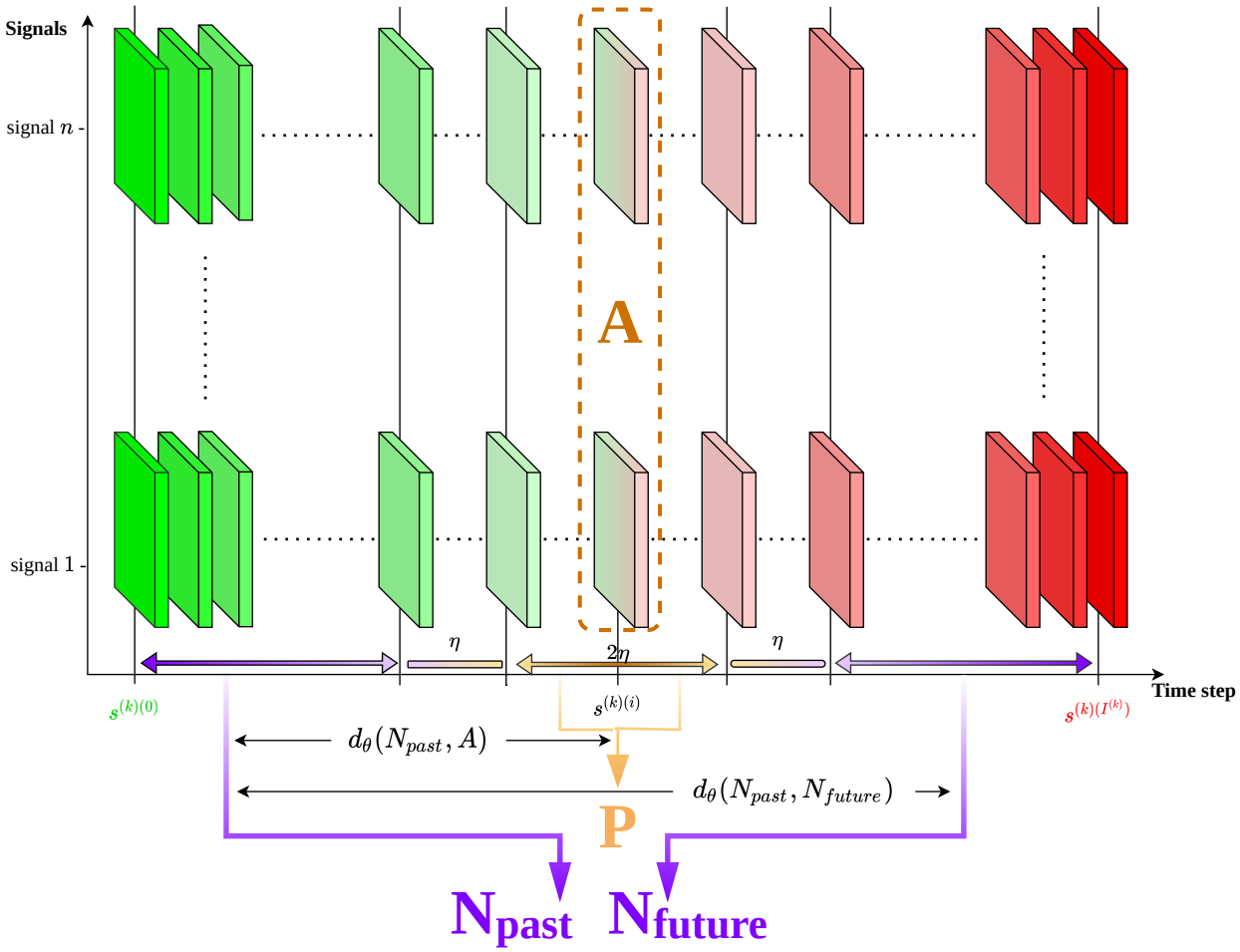


Figure IV.7: Double negatives strategy for adapting triplet loss to HI construction.

IV–2.1.4 Additional considerations

(a) Adaptive margin implementation

As mentioned previously in Subsection IV–1.3, the triplet loss with adaptive margin m should be preferred if one can provide information on how the positive and negative samples are dissimilar from each others, as it would improve the supervision given to the SNN's training. Here, the selection of triplets or quadruplets is guided by their degree of closeness in time. Hence, we propose an adaptive

Algorithm 3 Double-negative quadruplet selection for **one epoch**.

n is the number of signals $s^{(k)}$
S is the list of list of sub-signals $s^{(k)(i)}$ ▷ $S[k][i] = s^{(k)(i)}$
η is an integer and the neighbourhood hyperparameter
Quadruplets is an empty list of quadruplets
for $k \in 1 : n$ **do**
  $s^{(k)} \leftarrow S[k][:]$
  $I^{(k)} \leftarrow \text{len}(s)$
  for $i \in 1 + 2\eta : I^{(k)} - 2\eta$ **do**
    $a \leftarrow s^{(k)}[i]$
    $ind_p \leftarrow \text{random\_uniform\_pick}([i - \eta, i - 1] \bigcup [i + 1, i + \eta])$
    $ind_{n_p} \leftarrow \text{random\_uniform\_pick}([1, i - 2\eta])$
    $ind_{n_f} \leftarrow \text{random\_uniform\_pick}([i + 2\eta, I^{(k)}])$
    $p \leftarrow s^{(k)}[ind_p]$
    $n_p \leftarrow s^{(k)}[ind_{n_p}]$
    $n_f \leftarrow s^{(k)}[ind_{n_f}]$
    $Quadruplets.\text{insert}((a, p, n_p, n_f))$
  end for
end for

margin for a given triplet sample (A, P, N) where r and f in Eq. (IV.12) are expressed as follows:

$$r(s_i^{(k)}, s_j^{(k)}) = |i - j| \tag{IV.17}$$

$$f(r, (A, P, N)^{(k)}) = \frac{r(P^{(k)}, N^{(k)})}{I^{(k)}} \tag{IV.18}$$

where $(A, P, N)^{(k)}$ is a triplet from $\{s^{(k)(i)}(t)\}_{k \in \mathcal{K},\, i \in \mathcal{I}_k(T^{(k)}),\, t \in \mathcal{T}_{ki}}$. The relevance score r and the margin function f indicate that the further apart P and N are in their recording time, the further apart they should be in the SNN embedding space, thus the larger the margin m should be.

(b) HI denoising

To improve the HIs obtained with the proposed methodology, a denoising step can be applied on the HI curves $HI(t)$ to extract the trend, hereafter noted $T_{HI}(t)$. Similarly to the trend and residual decomposition performed for the robustness measure computation in Subsection III–1.1, LOESS is used for this denoising step. However, a slightly modified version of LOESS is here used with two differences:

(a) only neighbour points in the past of $HI(t)$ are used to estimate $T_{HI}(t)$ in contrast with the classical LOESS where neighbour points around $HI(t)$, both in the past and future, are used. The goal is to avoid $T_{HI}(t)$ values to be partially determined by future HI values, which in a context of prognostics for real-time cases would not be possible, and thus would overestimate the real performance of the proposed approach.

(b) the window length wl used for LOESS in Subsection III–1.1 for the robustness measure is set to five percent of the total HI length. However, here, for similar reasons as **(a)**, it is not possible to know in advance the total length of a HI, therefore, in a similar spirit, for each t, wl is set to five percent of the number of past HI values, i.e. $0.05t$. This implies a weaker smoothing at the beginning of the HI, but as t increases, so is the quality of the smoothing.

IV–3 Experiments

The previous sections of this chapter has laid out in details the proposed machine learning based methodology for constructing a model of health indicator given monitored measurements as MTS. This method has the ability to constrain monotonicity during the model learning without making any other assumptions on the HI trajectory evolution, e.g. linear or exponential. The proposed method, based on the contrastive learning theory, and more particularly the combination of SNN and contrastive losses, is a core contribution of this thesis to the field of PHM. In the following sections, this method is applied on two datasets widely used in the PHM community, usually for RUL prediction.
Firstly, Subsection IV–3.1 presents the experiment based on the Turbofan dataset, originally published in [166]. This dataset provides a relatively high amount of MTS issued from aircraft engine simulated with the C-MAPSS tool. C-MAPSS ("Commercial Modular Aero-Propulsion System Simulation") is a software for the simulation of realistic large commercial turbofan engine data. Because of the availability of a relatively large number of system instances monitored data, this experiment is focused on highlighting the efficiency of the method for enforcing monotonicity of the HI. Especially, proving that the two improved strategies (**anchor-at-start** and **double-negatives**) are better than the **naive** strategy.
In Subsection IV–3.2, the second experiment uses the bearing dataset originally published in [135]. This dataset provides one with a reduced number of instances of MTS issued from bearing undergoing an accelerated degradation. This dataset, although it presents only two dimensions in its MTS, is far more complex than the previous one, for its high-frequency sampling and the length of its generated MTS.

This experiment is rather focused on comparing the two improved strategies on a more challenging problem than that presented in Subsection IV–3.1. Additionally, a comparison is performed against a proposed method in the literature, that aimed at the same objective of enforcing monotonicity during the learning of a ML model, and without any additional assumptions on the evolution of the HI trajectories [150]. This latter is, to the author's knowledge, and at the time of writing of this chapter, the only attempt in the literature to build a HI model with this particular aim.

IV–3.1 Turbofan

The turbofan engine dataset provides monitoring data from simulated turbofans issued from the CMAPSS simulation software. It has originally been published in [166]. is divided into 4 different subsets of increasing difficulty. Details on the number of engines monitored, number of operating conditions and failure modes are given for each dataset in Table IV.3. In the simulation of the

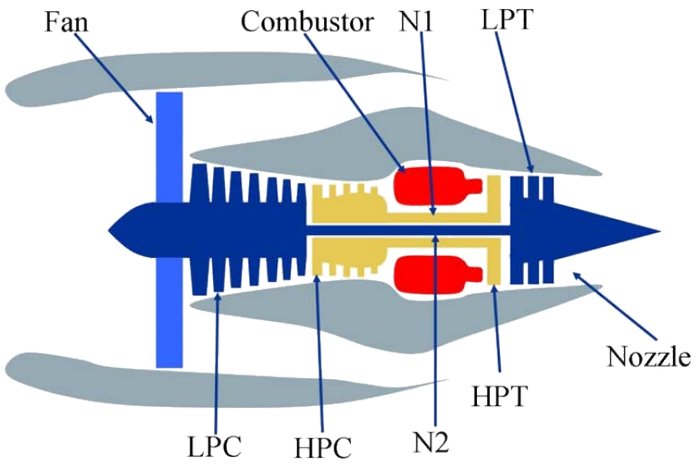


Figure IV.8: Illustration of the turbofans simulated by CMAPSS

engines the following parameters are used as inputs: Fuel flow, fan efficiency, fan flow modifier, fan pressure-ratio, low pressure compressor (LPC) efficiency, LPC flow, LPC pressure-ratio, high pressure compressor (HPC) efficiency, HPC flow, HPC pressure-ratio, high pressure turbine (HPT) efficiency, HPT flow and LPT efficiency. The outputs summarised in Table IV.1 are generated. Finally the operating conditions were controlled by the parameters summarised in Table IV.2

For each engine, there are thus 21 monitored signals and 3 operating condition signals, for a total of 24 dimensions in the multivariate time series, with a sampling frequency of one value per flight simulation. The monitored signals all represent a specific physical measure of the turbofan engine, e.g. temperature or pressure at critical points of the engine, core or fan speed. For the multiple operating condition subsets (FD002 and FD004), one of the six available operating conditions is randomly affected for each engine flight simulation, i.e. one per time step for each engine.
In our analysis we select the third dataset (FD003), where it is possible to test not only the HI construction method, but also for later exploring the HI-based failure model clustering thanks to the existence of two failure modes, see Chapter V.

IV–3.1.1 Preprocessing

In order to balance the importance of each of the 24 variables, a standard normalisation is first performed for each variable. As presented in Subsection IV–2.1, for each system instance, the entire multivariate time series $\{s^{(k)}(t)\}_{k\in\mathcal{K},\, t\in\mathcal{T}_k(T^{(k)})}$ where $\mathcal{K}=\{1,\dots,n\}$ and $n=100$ have to be divided into sub-signals $\{s^{(k)(i)}(t)\}_{k\in\mathcal{K},\, i\in\mathcal{I}_k(T^{(k)}),\, t\in\mathcal{T}_{ki}}$ of size l. The sub-signals $s^{(k)(i)}(t)$ are then considered as the sample units for the triplet selection. For this purpose, a sliding window operation with a sliding step $\Delta_G = 1$ and a window length $\Delta_L = 15$ steps is performed here. The value of 15 steps has been obtained by trial and error by the author: higher values ($[20, 30, 50]$) did not result in a significant improvement of the results while lower values ($[5, 10]$) did hinder these latter. A schematic view of this operation is given in Figure IV.4. No additional preprocessing step is applied here.

Symbol	Description	Unit
T2	Total temperature at fan inlet	°R
T24	Total temperature at LPC outlet	°R
T30	Total temperature at HPC outlet	°R
T50	Total temperature at LPT outlet	°R
P2	Pressure at fan inlet	psia
P15	Total pressure in bypass-duct	psia
P30	Total pressure at HPC outlet	psia
Nf	Physical fan speed	rpm
Nc	Physical core speed	rpm
epr	Engine pressure ratio (P50/P2)	-
Ps30	Static pressure at HPC outlet	psia
phi	Ratio of fuel flow to Ps30	pps/psi
NRf	Corrected fan speed	rpm
NRc	Corrected core speed	rpm
BPR	Bypass Ratio	-
farB	Burner fuel-air ratio	-
htBleed	Bleed Enthalpy	-
Nf_dmd	Demanded fan speed	rpm
PCNfR_dmd	Demanded corrected fan speed	rpm
W31	HPT coolant bleed	lbm/s
W32	LPT coolant bleed	lbm/s

Table IV.1: Outputs produced by CMAPSS during the turbofans simulation

Description	Min.	Max.	unit
throttle resolver angle (TRA)	20	100	degree
Mach number	0	0.84	-
Altitude	0	42000	ft.

Table IV.2: Operating condition parameters in CMAPSS during the turbofans simulation

Data-set	# runs	# operating conditions	# failure modes
FD001	100	1	1
FD002	260	6	1
FD003	**100**	**1**	**2**
FD004	249	6	2

Table IV.3: Turbofan dataset.

IV–3.1.2 Siamese neural network

In this application case, because input samples are MTS, the core NN of the SNN is firstly featured with recurrent layers for processing the sequential dynamics of the time series. Secondly, dense layers are stacked for compressing the samples into vectors of size e in the embedding space. The number of neurons for each new recurrent layer is set as the double of its input dimension, starting with 48, twice the dimension of the input samples. The number of recurrent layers n_r is left as an hyperparameter of the model. The recurrent layer implementation can come in different flavours, the most used ones being **SimpleRNN**, **LSTM** and **GRU**. While it is expected that **GRU** and **LSTM** obtain better performances than **SimpleRNN** in cases of longer time series, the relatively short ones used here (15 steps) might not necessitate the use of the more computationally demanding recurrent cell (**GRU** and **LSTM**). The type of recurrent layer is nevetheless considered as an hyperparameter here. The transition between recurrent and dense layers is **many-to-vector**, with every neuron output of every time step of the last recurrent layer flattened as an input vector for the first dense layer. The size of the first dense layer input and last dense layer output, are respectively determined by the last recurrent layer and the hyperparameter e, which is the size of the embedding space. To obtain the number N_h of neurons per hidden dense layers, we use the formula proposed in [123]:

$$N_h = \lfloor \sqrt{N_i N_o} \rfloor \tag{IV.19}$$

where N_i and N_o are respectively the input and output sizes of a dense NN with a single layer. Similarly, this formula is adapted to fit networks with more hidden layers in our work:

$$N_{h;k} = \lfloor (N_{h;k-1})^{1-\frac{1}{n_l-k+2}} \cdot (N_o)^{\frac{1}{n_l-k+2}} \rfloor, \; k \in [1, \ldots, n_{hd}] \tag{IV.20}$$

where $N_{h;0} = N_i$ is the input size and n_{hd} is the number of layers. The number of hidden dense layers n_{hd} in the SNN core architecture is a hyperparameter of the model.

IV–3.1.3 Training

The models are trained on 80 out of the 100 available turbofan runs in the dataset. The 20 leftover ones are used as both the validation and testing set. The training is stopped after 50 epochs without any improvement of the loss on the validation set. The training is performed via a stochastic gradient optimisation algorithm (a.k.a. optimiser) called **adam** and presented in Chapter II. Two hyperparameters are crucial to fine-tune for this optimiser, the batch size n_B and the learning rate r. The batch size is simply the number of samples inside a batch, and the learning rate is the amplitude of the parameters' adjustment in the direction of the loss gradient.

IV–3.1.4 Hyperparameters fine-tuning

The values of the hyperparameters summarised in Table IV.4 are optimised via different grid searches on all three strategies, **naive**, **double-negative** and **anchor-at-start**. The hyperparameters to be optimised during this fine-tuning as well as the different values tried during the grid search are also summarised in the table.
Because the number of hyperparameters is quite high, a grid search on all the hyperparameters

combined would lead to a very high number of trials, with one complete SNN training (until convergence of the loss) per trial:

Naive : $3^7 = 2187$ trials

Anchor-at-start : $5 \times 3^7 = 10935$ trials

Double-negative : $5 \times 3^7 = 10935$ trials

For a total of: 24057 trials

To drastically reduce the number of trials, several hypotheses are made. The first hypothesis is that the hyperparameters of the core neural network, are not dependent on the other hyperparameters, i.e. the optimiser hyperparameters, the hyperparameter of samples selection and the hyperparameters relative to the strategy for enforcing monotonicity. Secondly, the optimiser hyperparameters are not dependent on the hyperparameter of samples selection and the hyperparameters relative to the strategy for enforcing monotonicity. It results from these two hypotheses that for each of the three strategies, the fine-tuning is divided in three grids. First, the core NN hyperparameters (4 hyperparameters with 3 possible values) are optimised via a grid search with all the other hyperparameters fixed at any values that will be later tested. Then the optimiser parameters (2 hyperparameters with 3 possible values) are optimised with all the other parameters fixed (the core NN hyperparameters are fixed with their previously found optimal values). Finally, the hyperparameter of samples selection and the hyperparameters relative to the strategy for enforcing monotonicity are optimised, which results in the following number of trials:

Naive : $3^4 + 3^2 + 3 = 93$ trials

Anchor-at-start : $3^4 + 3^2 + 3 \times 5 = 105$ trials

Double-negative : $= 3^4 + 3^2 + 3 \times 5 = 105$ trials

For a total of: 303 trials

During the grid searches, all the combinations of all the possible values for each hyperparameter are tried by training a model for each of these combinations, following the training procedure explained in Subsection IV–3.1.3. Usually, during a fine-tuning, the criteria for selecting the best hyperparameters is the loss on the validation test. Here, however, we are more interested in the monotonicity criterion of the obtained HI. For each trained model, all the HI trajectories for the validation set are computed and the mean monotonicity values for all these trajectories is used as the criteria for selecting the best hyperparameters values. Later on in this section, the link between the loss and the final monotonicity of HI is studied.

The fine-tuning for each strategy yield a similar architecture represented in Figure IV.9, the only difference being the type of recurrent neuron cell. The **double negative** and **naive** strategies obtain their best results with simple RNN cells, while the **anchor at start strategy** performs better with LSTM cells. The rest of the hyperparameters values can be found in Table IV.4

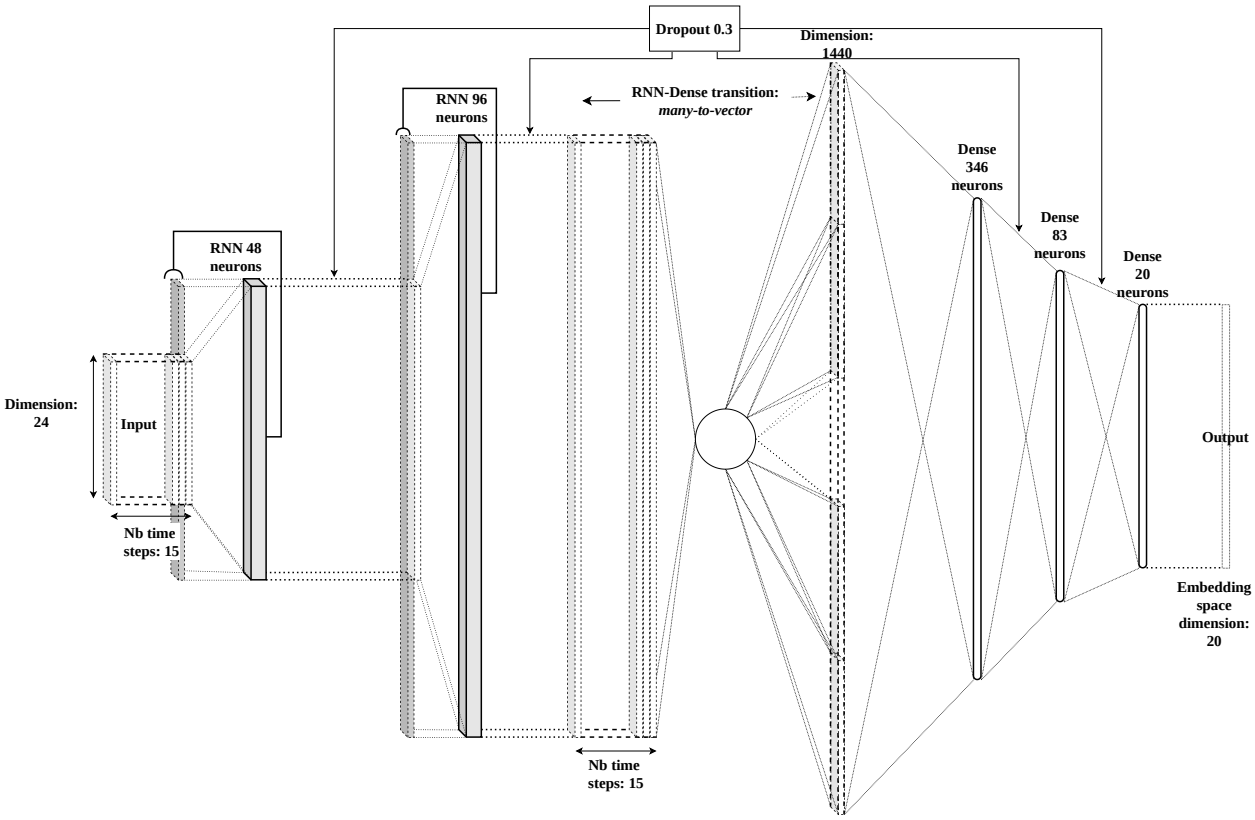


Figure IV.9: Core NN architecture for the Turbofan dataset with optimised hyperparameters.

IV–3.1.5 Final training for performance evaluation

Once the hyperparameters values are found, for each SNN-based strategy, five runs with the same hyperparameters are trained. The HI trajectories of the training and testing sets are then computed. The HI performance criteria are subsequently computed on the training and testing trajectories separately for each run, and then averaged for each strategy. The use of multiple runs is made for counter the randomness arising from the weight and biases initialisationn and the inherent random triplet selection strategies. More than five runs could have been performed for each strategy, but would have required significantly more computing time. It turned out that the randomness arising from parameters initialisation and triplet selection does not seem to hinder the conclusion we are drawing from this experiment.

IV–3.1.6 Performance evaluation and discussion

(a) Loss and monotonicity evolution during the learning

We first analyse how losses and monotonicity values respectively assessed on the training and testing sets evolve w.r.t. the number of epochs.
To do so, for each SNN based strategy, i.e. **naive**, **anchor-at-start** and **double-negative**, and for each run, the loss and monotonicity values are recorded for each epoch whilst optimising the parameters θ of the SNN-based models. If our proposed approach works in aiming at constraining the monotonicity, a correlation should be observed between a decreasing loss and an increasing

Hyperparameters			Trial values	Optimal values		
				Anchor at start	Double negative	Naive
Model	Core NN	e: SNN embedding size	5,10,20	20		
		n_{hd}: number of hidden dense layers	[1-3]	2		
		n_r: number of recurrent layers	[1-3]	2		
		Type of recurrent layers	GRU, LSTM, Simple RNN	LSTM	Simple RNN	Simple RNN
Optim.	Optimiser	r: Optimiser learning rate	0.01, 0.001, 0.0001,	0.0001		
		n_B: Batch size	16,32,64	64		
	samples selection	η: Temporal neighbourhood for triplet selection	5,10,15	10		
	Anchor-at-start	λ: semi-hard margin	[1-5]	1	-	
	Double-negative	m_{pen}: loss penalisation margin	0.1, 0.3, 0.5, 0.7, 1	-	0.3	-

Table IV.4: Hyperparameters of the method optimised for the turbofan dataset.

monotonicity. To this end, loss and the opposite monotonicity values are then averaged over the five runs and drawn w.r.t. a time index expressed in percentage of the total amount of epochs. Before being drawn, their respective means and standard deviations are used to normalise, or center-reduce, the values. Results are plotted on Figure IV.10, left-end side.

(1) It is first expected that losses decrease with the number of epochs, a proof that the model is learning. It is nevertheless also expected that the validation loss start to stagnate or increase after a certain number of epochs, thus triggering the end of the training. This is exactly what is being observed in Figure IV.10 for the three proposed strategies, only proving that the training works as expected.

(2) Since our proposed SNN-based strategies are designed to enforce monotonicity, we should observe that the loss of the model is correlated with the monotonicity criteria. For a decreasing loss, we thus expect an increasing monotonicity, and thus a decreasing opposite monotonicity. The observations once again come to corroborate these expectations. However, the **anchor-at-start** strategy and even more significantly the **double-negative** strategy seem to perform better than the **naive** one.

(3) To analyse in details these performance differences, the evolution of the Pearson correlation between loss and the opposite of the monotonicity criterion is plotted on the right-side of Figure IV.10 for each SNN-based strategy versus the number of epochs. More precisely the curves of correlation between the following pairs of values are plotted: (training loss, testing loss), (training loss, -training monotonicity), (training loss, -testing monotonicity). Each point of a correlation curve indicates the current correlation between the values up to this point.

From these plots, it is clear that the naive strategy does not clearly lead an improvement in the loss to be correlated with an improvement in monotonicity, except at the very beginning of the training. When reaching 40% of total training time, we can even witness a clear inverse correlation where the loss is slightly improving while the monotonicity is drastically worsened. For the two other strategies, however, the loss and monotonicity are significantly correlated. It is particularly interesting to see that for the **anchor-at-start** strategy, the monotonicity on the testing set seems more correlated to the loss on the same set than the monotonicity on the training set and vice versa, especially at the end of the training when an overfitting phenomenon is observed. For the **double-negative** strategy, the correlation between the loss and monotonicity is unquestionable during the entire training, except on a short-phase during the middle part of the training. For the **anchor-at-start** strategy, if the correlation is also high, a short period of decorrelation can be witnessed at the beginning of the training. A general conclusion of these observations is that the two strategies for enforcing monotonicity proposed here are effective, with a clear advantage for the **double-negative** strategy.

(b) HI performance criteria

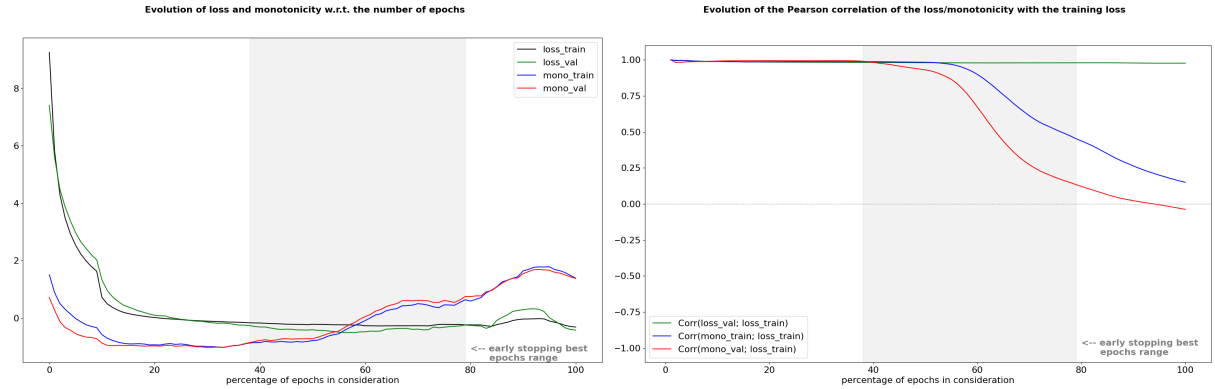
For better visualisation, only HI curves of the testing data-sets obtained with the run having the best monotonicity value for each of the three proposed strategies are represented in Figure IV.11. However, Table IV.5 gives the associated performance measures averaged over all the five runs on both the training and testing sets for the three strategies.

Dataset	Strategy		Mon	Rob	Pro	Fco
Training	Naive	Raw	0.770	0.123	0.739	0.651
		Denoised	0.774	0.104	0.736	0.657
	Anchor at start	Raw	0.820	0.207	0.752	0.910
		Denoised	0.831	0.162	0.754	**0.922**
	Double negative	Raw	0.846	0.109	**0.797**	0.911
		Denoised	**0.851**	**0.090**	0.793	0.902
Testing	Naive	Raw	0.725	0.125	0.782	0.708
		Denoised	0.728	0.105	0.778	0.724
	Anchor at start	Raw	0.805	0.162	**0.859**	0.901
		Denoised	0.812	0.130	0.858	0.914
	Double negative	Raw	0.850	0.114	0.841	**0.956**
		Denoised	**0.855**	**0.095**	0.842	0.954

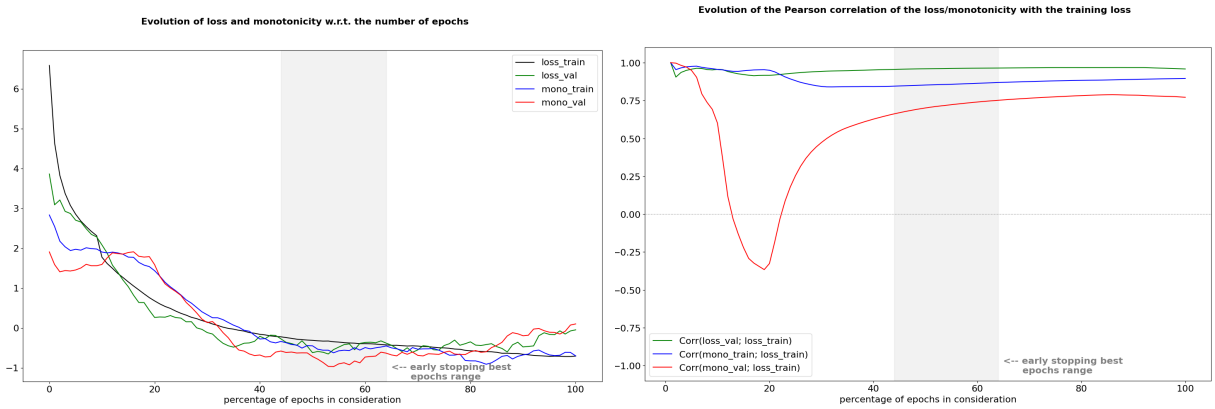
Table IV.5: Performance indicators on turbofan dataset. Monotonicity and robustness are averaged over their respective (training, testing) sets.

From the results in Table IV.5, one should note that the **double-negative** strategy performs slightly better than the **anchor-at-start** one which in turn performs better than the **naive** strategy. In particular, on the testing dataset, the denoised HIs of the **double-negative** strategy reach the best scores on monotonicity and robustness, while their raw HIs have the best failure consistency. Anchor-at-start strategy only outperforms **double-negative** for the prognosability criterion

(a) Naive



(b) Anchor-at-start



(c) Double-negative

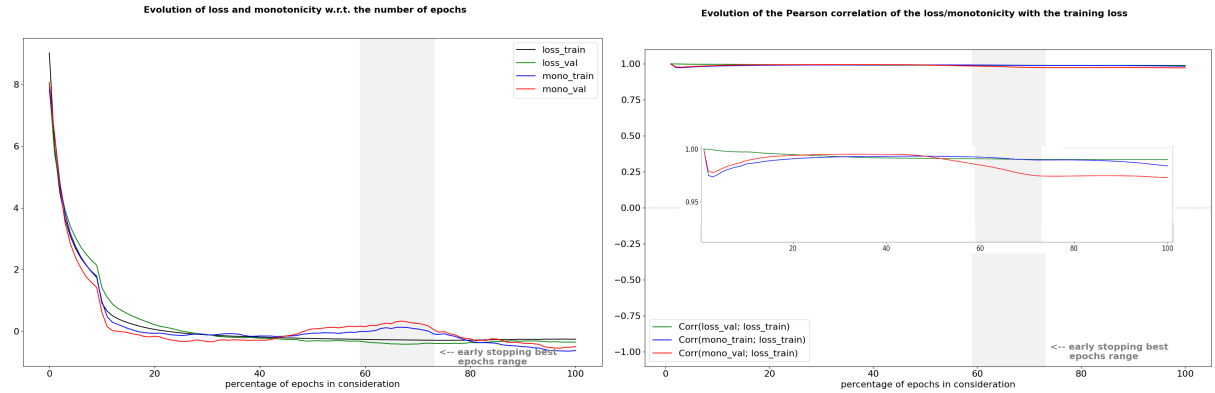


Figure IV.10: Graphs issued from the analysis of loss and monotonicity evolution during the learning

on the testing set. Regarding monotonicity, **anchor-at-start** and **double-negative** strategies lead to a value higher than 0.8. More generally, **anchor-at-start** and **double-negative** strategies are systematically better than the **naive** strategy in terms of monotonicity. These monotonicity values match with the findings obtained on the previous analysis. Moreover they can be connected to the obserbed HI trajectories in Figure IV.11 where for the **naive** strategy, long HI plateaus can be observed at the end of life of turbofans hence hindering monotonicity critarion. These plateaus are far less obvious on the HI trajectories obtained with **anchor-at-start** and almost nonexistent with

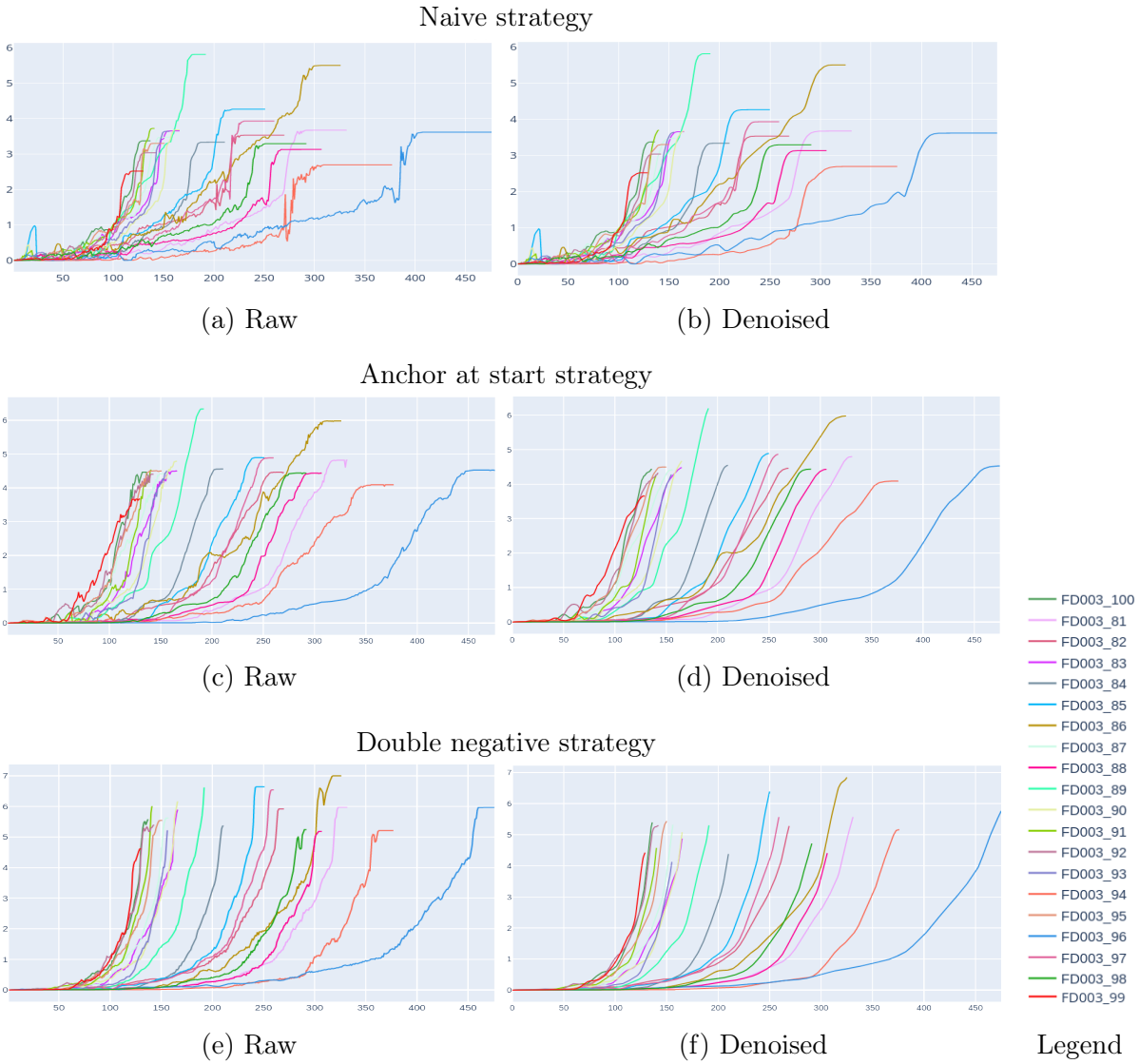


Figure IV.11: HI curves on turbofan testing dataset (20 out of 100 turbofan runs), the x-axis and y-axis represent the time step and the HI value, respectively.

double-negative. This result indicates that the proposed methods for enforcing monotonicity in these two strategies are efficient, with still a net advantage for the **double-negative** strategy.

For failure consistency measures, values higher than 0.9 are obtained for **anchor-at-start** and **double-negative** strategies, and are significantly lower for the **naive** strategy, once again showing the superiority of the two improved strategy. This indicates that the HI curves have values at the end-of-life that are similar.

For prognosability **anchor-at-start** and **double-negative** are also significantly higher than the **naive** strategy, which indicates that for these two improved strategies, curves from different instances of the same systems have similar evolutions.

Figure IV.12: Image of bearings issued from the experimental platform of [135] and their degradations

Finally, with values of robustness measurement close to 0.1, these HIs are only blurred with a relative percentage error accounting for about 10% of the HI values in average. An interesting observation is that the denoising step seems to often improve the score of the HI curves for each performance measures, and these latter are never significantly lower after the denoising step.

All of these observations indicate that the RUL prognostics based on these HIs could be considerably eased. These good performance results can be directly visualised from the test HI curves obtained with the single best model for each strategy plotted in Figure IV.11. HI curves constantly increase, are of similar shapes and reach similar values at the end of lives of turbofan especially for the **double-negative** strategy.

IV–3.2 Bearing

The Femto bearing dataset [135] consists of bi-variate time series $\{s^{(k)}(t)\}_{k\in\mathcal{K},\, t\in\mathcal{T}_k(T^{(k)})}$ with $s^{(k)}(t) \in \mathbb{R}^d$, $d = 2$, measuring horizontal and vertical vibration signals recorded by accelerometers on $n = 17$ rolling bearings subjected to different operating conditions until failure. The experimental platform is illustrated in Figure IV.13. It controls the operating conditions which are combinations of the rotating speed of the bearing inner ring compared to the fixed outer ring, handled by the rotating module (yellow in Figure IV.13), and the radial force applied on the outer ring, handled by the load module (blue in Figure IV.13). Images of the tested bearings can be found in Figure IV.12. The details on the operating conditions among the bearings are given in Table IV.6. The time series of bearing vibrations are directly recorded in the form of sub-signals $\{s^{(k)(i)}(t)\}_{k\in\mathcal{K},\, i\in\mathcal{I}_k(T^{(k)}),\, t\in\mathcal{T}_{ki}}$ where $s^{(k)(i)} \in \mathbb{R}^2 \times \mathbb{R}^{2560}$ thus of length $\Delta_l = 2560$ corresponding to 100msec sampled at 25.6kHz, each sub-signal spaced by $\Delta_G = 10$sec.

(a) Dataset specific difficulty

As opposed to the previous dataset, this one poses great challenges due to several factors:

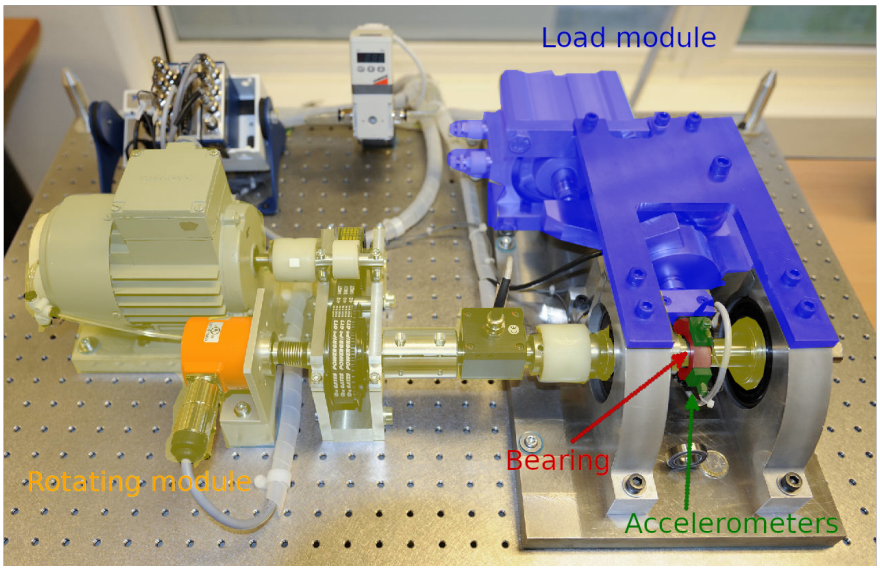


Figure IV.13: Image of bearings issued from the experimental platform of [135]

Condition	Rotating speed (rpm)	Radial force (N)	Bearings
1	1800	4000	1_1, 1_2, 1_3, 1_4, 1_5, 1_6, 1_7
2	1650	4200	2_1, 2_2, 2_3, 2_4, 2_5, 2_6, 2_7
3	1500	5000	3_1, 3_2, 3_3

Table IV.6: Femto bearing dataset.

Only 17 run-to-failures are available. This is a very low amount for generalising degradation trajectories for this type of bearing. Moreover, these bearings have different operating conditions, which increase the potential differences in terms of degradation kinetics and degradation scenarios between each run-to-failure.

The length of run-to-failures are greater and less equally distributed than for the previous dataset. For comparison, the turbofan datasets has lengths varying from 150 to 500 time steps for the raw MTS, while the bearing dataset has lenghts varying from 200 to 2800 sub-signals. This fact combined with the low number of run-to-failures lead to a particularly complex situation where of the number of samples is significantly lower than the dimension of the samples: 17 samples for dimensions varying from $200 \times 2560 \times 2$ to $2800 \times 2560 \times 2$, i.e. roughly one to fifteen millions, here 2 denotes for the number of signals, i.e. vertical and horizontal vibrations.

The high frequency sampling of 25.6kHz is also a factor that increase the complexity with inherently more potential noise in the data.

The inherent complex dynamic of bearings . This point is from our experience the most important and irreducible challenge of this application case. Rolling bearings are known to be very complex systems with a multitude of potential failure modes. Bearing degradations are often triggered by grease ageing. And the associated failure modes can include; cage failure, abrasive of adhesive wear, indentation of spalling. This dataset does not provide with the failure modes associated to each bearing that could be obtained with post-mortem inspections of the failed bearings.

IV–3.2.1 Preprocessing

For this experiment, the data is directly given in the form of sub-signals $\{s^{(k)(i)}(t)\}_{k\in\mathcal{K},\, i\in\mathcal{I}_k(T^{(k)}),\, t\in\mathcal{T}_{ki}}$ and the complete time series $\{s^{(k)}(t)\}_{k\in\mathcal{K},\, t\in\mathcal{T}_k(T^{(k)})}$ are therefore unavailable. Hence, the sliding window operation is not necessary, as it has already been established by the experiment. In contrast with the previous dataset, the number of time steps l per sub-signals is much larger because of the high sampling frequency, i.e. $l = 2560$. Moreover, it is expected that the high frequency content is of crucial importance for estimating the bearing health state. In order to further ease the learning of the SNN model, a two-step preprocessing is proposed in this work.

(a) Raw signal denoising

A first step of wavelet denoising is proposed with a discrete Meyer mother wavelet ψ, as this procedure has already proven to be efficient for denoising high frequency time series of bearing vibrations [1].

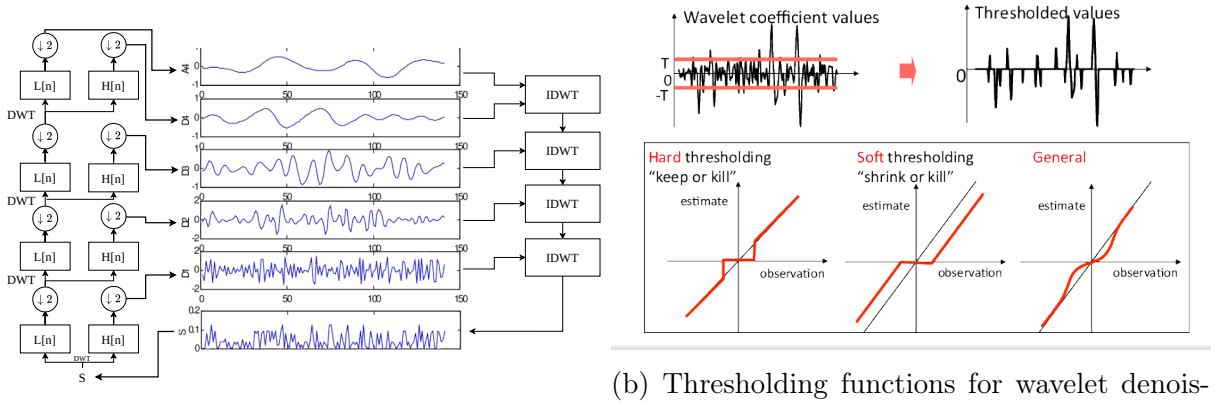
Discrete wavelet denoising consists in repeatedly applying a band-pass filter and a low-pass filter to the signal, thus dividing the signal into several sub-signals, each corresponding to a certain frequency band. The band-pass filter at each level j of the decomposition is a level-scaled wavelet function ψ_j and the low-pass filter is the corresponding level-scaled scaling function ϕ_j, see Figure IV.14a for a schematic overview. At each level j of decomposition, the result of the band-pass filter and the low-pass filter are:

$$H(t) = \sum_{k=-\infty}^{+\infty} s(k)\psi_{j;k}(2t-k) \tag{IV.21}$$

$$L(t) = \sum_{k=-\infty}^{+\infty} s(k)\phi_{j;k}(2t-k) \tag{IV.22}$$

where $\psi_{j;k}$ corresponds to the j-scaled wavelet function shifted by k defined as $\frac{1}{\sqrt{2^j}}\psi(\frac{t-k2^j}{2^j})$ and $\phi_{j;k}$ is the corresponding scaling function for the same level j and shift k. $s(k)$ at the level 0 is the signal to denoise and at the level j it is the result of the preceding low-pass filter $L_{j-1}(t)$. Once the signal is decomposed to its maximum decomposition level $J = \lfloor \log_2(\frac{len(\text{signal})}{len(\psi(t))-1}) \rfloor$, a thresholding function is applied to each sub-signal in order to remove or quiet the values below the threshold, as represented in Figure IV.14b.
The thresholding function and threshold value are set using the BayesShrink thresholding method [25]. Denoised sub-signals per frequency band are thus obtained as shown in Figure IV.14b. Fi-

nally, the complete denoised signal is obtained by repeatedly applying the inverse discrete wavelet transform (IDWT) between the two denoised sub-signals with the lowest frequency bands until all sub-signals are merged back together, as shown on the right side of Figure IV.14a. Denoised versions of the raw vibration signals are thus obtained.



(a) Discrete wavelet transform and its inverse.

(b) Thresholding functions for wavelet denoising.

Figure IV.14: Wavelet denoising.

(b) Time-frequency representation

Because of the aforementioned high-frequency sampling of the sub-signals $\{s^{(k)(i)}(t)\}_{k\in\mathcal{K},\, i\in\mathcal{I}_k(T^{(k)}),\, t\in\mathcal{T}_{ki}}$, on top of the denoising operation, a preprocessing method able to extract complex time-frequency features is applied. Inspired by the work of [218, 71], we resort to the continuous wavelet transform (CWT) to turn our bi-dimensional sub-signals into two complex time-frequency image representations. The CWT consists of the convolutions of scaled versions of a mother wavelet, each scale corresponding to a certain frequency and each convolution step corresponding to a certain position in time. CWT thus produces coefficients $CWT_x(t,a)$ indicating how much amplitude the frequency f_a (corresponding to scale a) contains at time t. By applying this transform with J scales on each signal for every sample of each bearing, we obtain two matrices of size $l \times J$ per sample, one for vertical vibrations and one for horizontal vibrations. These matrices are composed of the coefficients $CWT_x(t,a)$ and can be represented as images, an example of an obtained image from CWT on a bearing signal vibration is given in Figure IV.15.

The continuous wavelet transform of a signal $x(t)$ can be written as follow:

$$CWT_x(t,a) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} x(\tau)\psi(\frac{\tau - t}{a})d\tau$$

where $\psi(t)$ is the mother wavelet, a the scale variable associated to a pseudo frequency $f_a = \frac{f_c}{a\Delta}$, f_c the central frequency of $\psi(t)$ and Δ the sampling frequency of $x(t)$. For each time t and for each pseudo-frequency f_a we thus obtain a coefficient $CWT_x(t,a)$ indicating how much amplitude the frequency f_a contains around time t. In order to choose the optimal scales $\{a_j\}_{j=0}^{J}$, where J is the

number of scales in the transform, we rely on the following rules derived from [191]:

$$a_0 = \frac{f_c}{f_{max}} \tag{IV.23}$$

$$a_J = \frac{f_c}{f_{min}} \tag{IV.24}$$

$$\delta_j = \frac{\log_2(\frac{a_J}{a_0})}{J} \tag{IV.25}$$

$$a_j = a_0 2^{j \cdot \delta_j}, \forall j \in (1, \ldots, J) \tag{IV.26}$$

where f_{max} and f_{min} are respectively the maximum and minimum frequencies scanned by the CWT. In order to reduce the computational burden of our model, the size of matrices to be obtained is

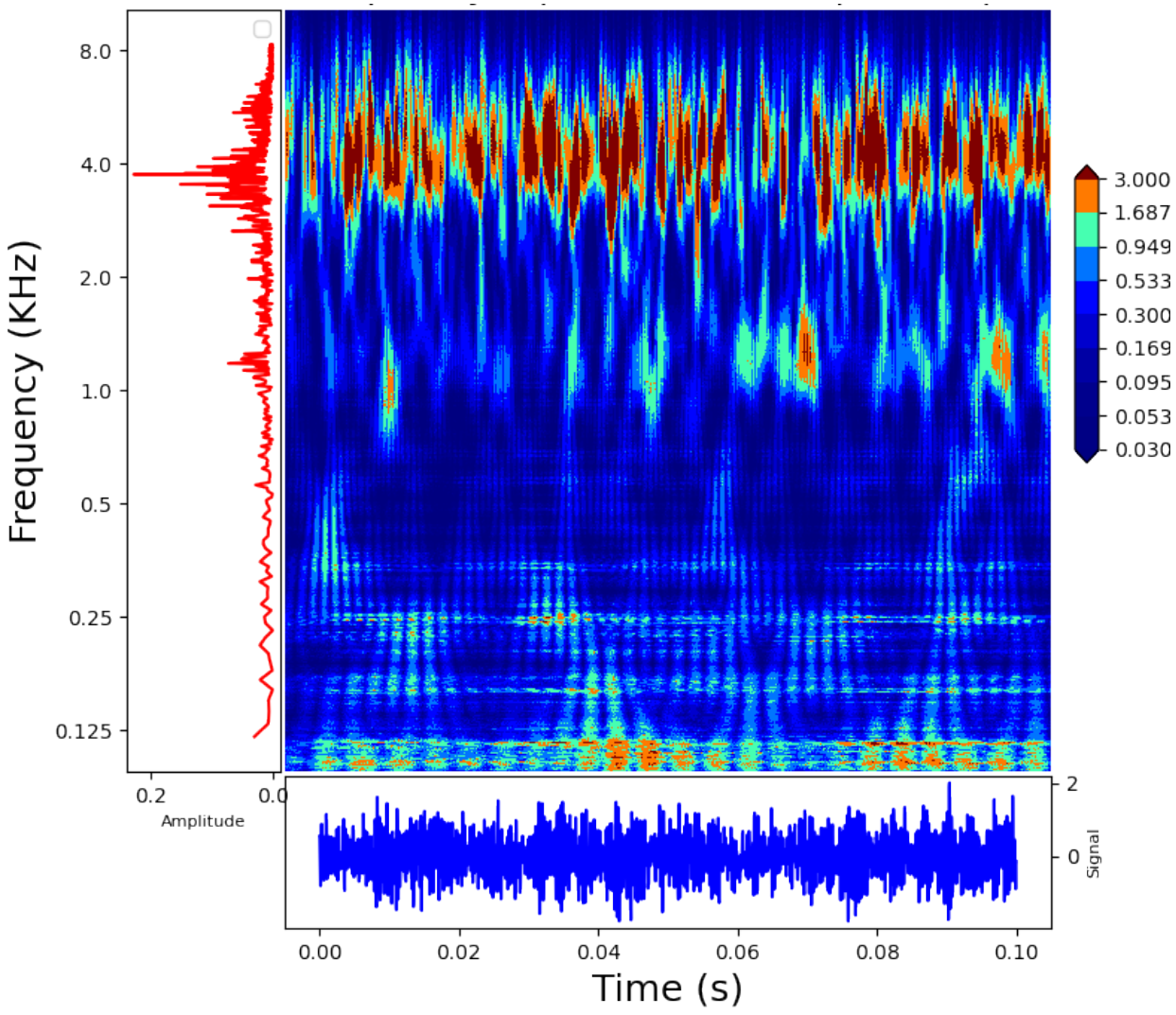


Figure IV.15: Illustration of a time frequency image obtained by CWT of bearing vibration signal.

fixed to 128×128 values. $J = 128$ was therefore chosen as the number of scales in the CWT. Matrices of 256×128 were therefore obtained, which were then transformed into compressed images of 128×128 grayscale pixels were the values of the pixels represents the CWT coefficient, the darker the pixel, the higher the CWT coefficient. It is worth mentioning that f_{max} and f_{min} values used

were 8500Hz and 20Hz, respectively. The maximum frequency was chosen manually by trial and error in order to remove the high frequencies that are mostly noisy. The minimum frequency was chosen so that it is slightly lower than the frequency of a full rotation of the bearing inner ring. The minimum rotating speed during the recording of the bearing vibration signals being of 1500 rotations per minute equivalent to 25 rotations per second, the minimum frequency of a full rotation of a bearing inner ring is therefore 25Hz. To illustrate this time frequency transform, the obtained images for all bearings at different percentage of their total life duration are shown in figs. IV.16 to IV.18. The analysis of these images can give interesting information. The main one is that for every bearing the images at the very end of life are significantly different compared to its previous images. Moreover, the order of magnitude of this difference is far superior to the previous ones. For the images prior to the final one, most bearing tend to show a more smooth degradation, with some fluctuations for some of them especially those with the second operating condition depicted in Figure IV.17. For the bearings with the first operating condition, represented in Figure IV.16 two bearings (1_3 and 1_4) seem to experience a sudden degradation prior to to their end of life (respectively at 70% and 80% of their total lives). It is thus expected to recover these observations in the HI trajectories that will be obtained on this dataset.

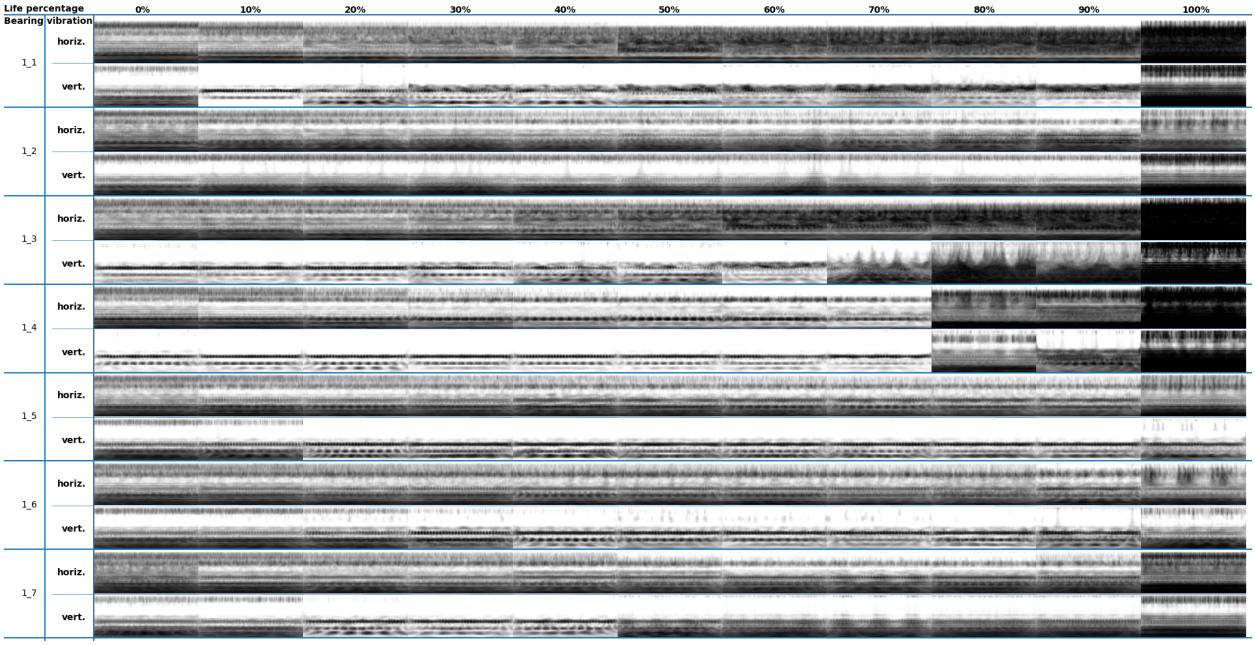


Figure IV.16: Obtained CWT time-frequency images at different percentage of total life of bearings in the **first** operating condition

IV–3.2.2 Siamese core neural network

A SNN principally featured with convolutional layers is herein implemented, since this type of layers is specially adapted for the processing of images. Dense layers are also added for compressing the outputs of the convolutional last layer into vectors in the embedding space. The convolutional architecture is composed of a succession of convolution blocks, themselves featured of a convolutional layer followed by a pooling layer. The use of these convolution blocks is a recommended practice for CNN [56] that has also been used by [218] and [71] for learning respectively a HI and RUL from

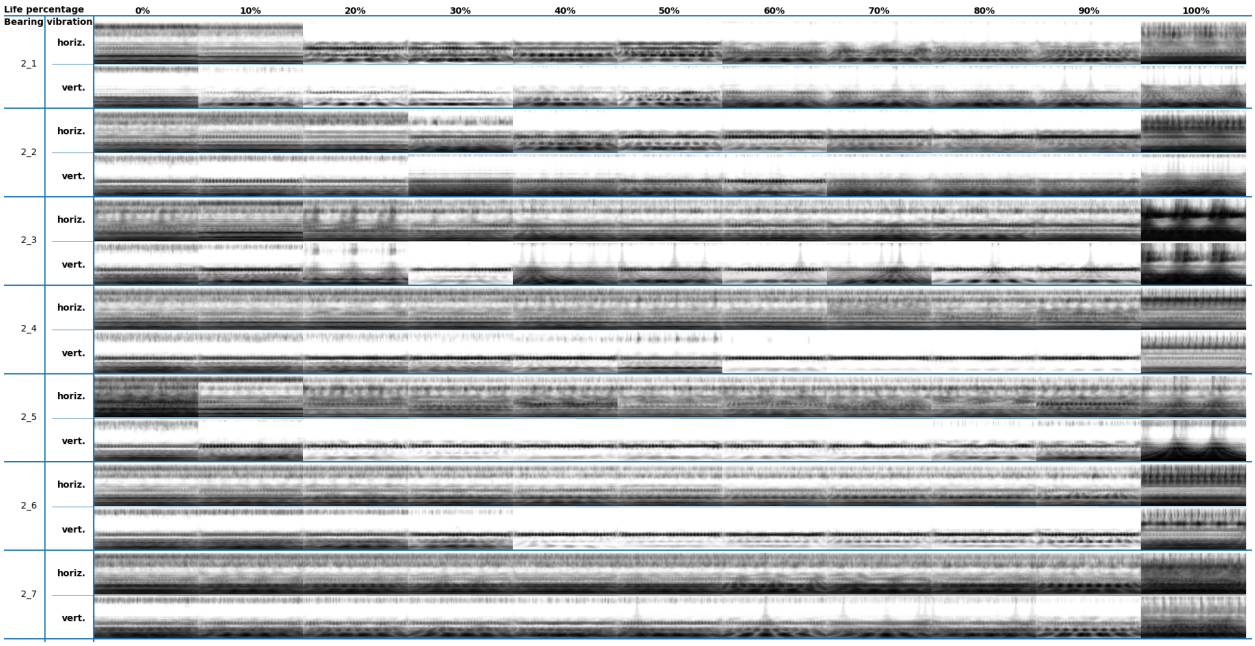


Figure IV.17: Obtained CWT time-frequency images at different percentage of total life of bearings in the **second** operating condition

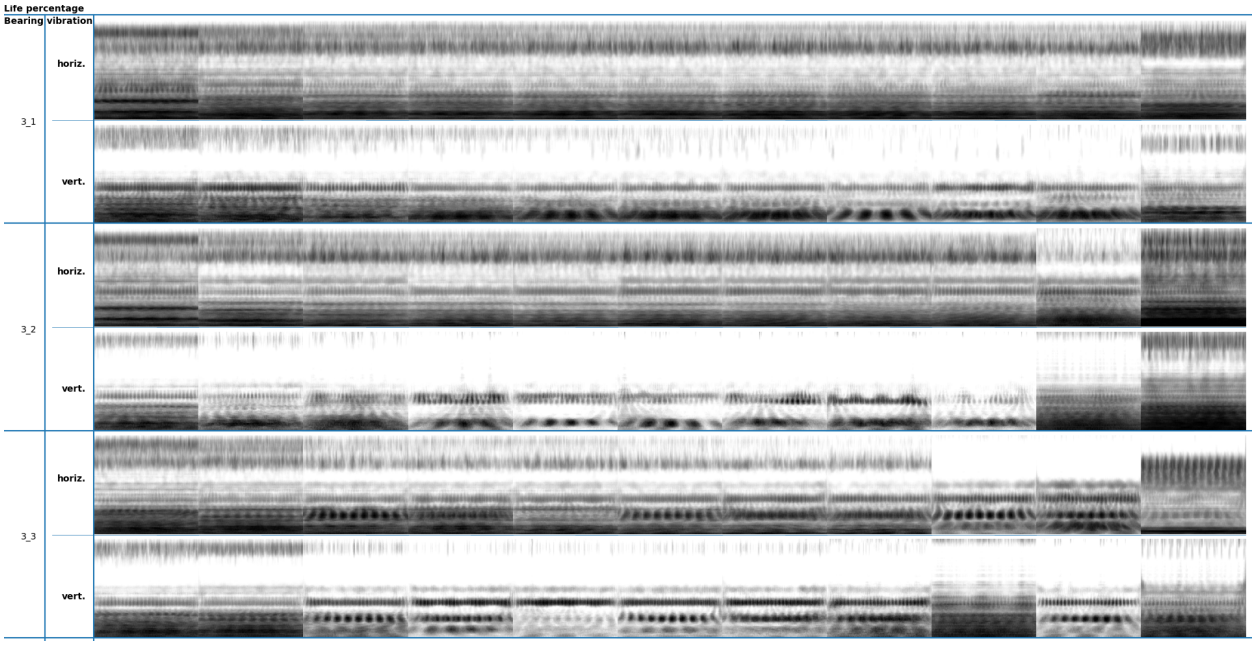


Figure IV.18: Obtained CWT time-frequency images at different percentage of total life of bearings in the **third** operating condtion

CWT images of bearing vibrations. The filter size of the pooling layers size is set to 2×2, while for the convolution layers it is kept to the default value of 3×3. The number of filters in the kernel at each convolution layer starts at 16 for the first block and is doubled for each subsequent block to compensate the halving width and height of the outputs of the pooling layers, as recommended in [56]. The outputs of the last convolutional block are then flattened into a vector before being fed

to the dense layers. Dense layer sizes are set as defined in Eq. (IV.20), similarly as for the previous experiments. The number of convolutional block n_c is an hyperparameter, as well as the number of final dense layers n_{hd}.

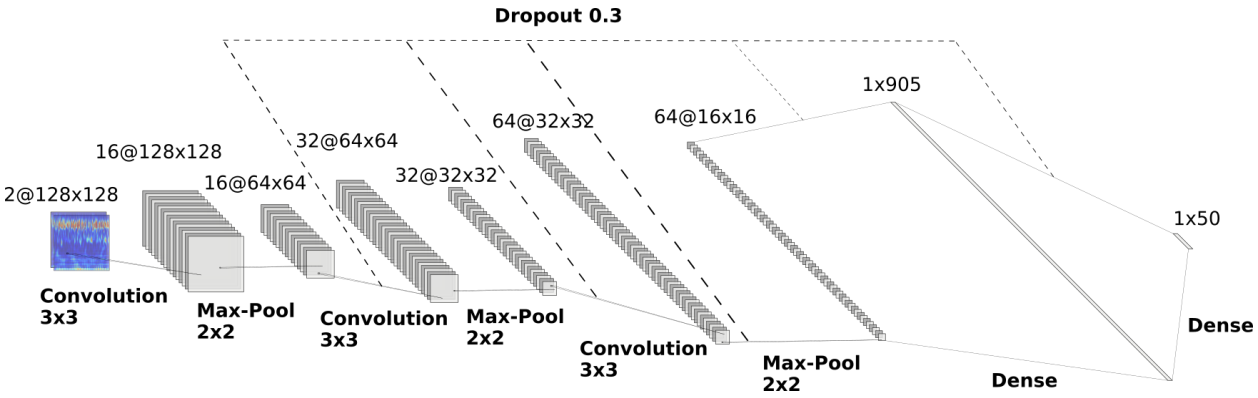


Figure IV.19: Core NN architecture for the bearing dataset.

IV–3.2.3 Training

For this dataset, which is affected by data scarcity and complex high frequency measurements, the training has been carried out in four different setups with increasing difficulty. For all these setups, building up from the previous results on the turbofan dataset, only the **anchor-at-start** and **double-negative** strategies are tested.

#**1** In this setup, the training is carried out on the entire available set, i.e. all the 17 bearings, and stopped when the training loss stops decreasing. There is therefore no validation and testing sets. This setup is only for testing the ability of the proposed strategy to correctly learn from this complex dataset.

#**2** Here, the model is trained using the leave-one-out strategy on the entire dataset. The model is therefore trained 17 times, with a single bearing left out of the training set for each run and used as validation and testing set.

#**3** Building up on the results from the previous setup, the setting proposed in [216] is used where 4 bearings for which the generalisation is performed well in the previous setup (1_1, 1_3, 2_6 and 3_3) are kept out of the training set and used for testing and validation sets.

#**4** Comparison with the VAEDC approach of [150]. This is the same setup as the previous one, except a new approach is used for comparison and thus trained similarly.

Except for the setup #**1**, the training is stopped after 50 epochs without any improvement of the loss on the validation set. The training is performed via the same stochastic gradient optimisation algorithm as for the previous dataset: **adam**. The batch size n_B and the learning rate η are again considered as hyperparameters with the experiments on this dataset

IV–3.2.4 Hyperparameter fine-tuning

As for the experiment on the turbofan dataset, the values of the hyperparameters summarised in Table IV.7 are optimised via different grid searches for the two tested strategies, **double-negative** and **anchor-at-start**. The hyperparameters to be optimised during this fine-tuning as well as the different values tried during the grid search are also summarised in Table IV.7.

The same procedure is used for the fine-tuning with a three-step grid search, considering the same hypotheses of hyperparameters dependance as in Subsection IV–3.1.4:

- Hyperparameters of the core neural network, that is the architecture of the neural network are not dependent on the other hyperparameters, i.e. the optimiser hyperparameters, the hyperparameter of samples selection and the hyperparameters relative to the strategy for enforcing monotonicity.
- The optimiser hyperparameters are not dependent on the hyperparameter of samples selection and the hyperparameters relative to the strategy for enforcing monotonicity.

Thus, first, the core NN hyperparameters (4 hyperparameters with 3 possible values) are optimised via a grid search with all the other hyperparameters fixed. Then, the optimiser parameters (2 hyperparameters with 3 possible values) are optimised with all the other parameters fixed (the core NN hyperparameters are fixed with their previously found optimal values). Finally, the parameter of samples selection and the parameters relative to the strategy for enforcing monotonicity are optimised, which results in the following number of trials:

Anchor-at-start : $3^3 + 3^2 + 3 \times 5 = 51$ trials

Double-negative : $= 3^3 + 3^2 + 3 \times 5 = 51$ trials

For a total of 102 trials

During the grid searches, all the combinations of all the possible values for each hyperparameter are tried by training a model for each of these combinations, following the training procedure explained in Subsection IV–3.1.3. Here, setup #**1** is used for the hyperparameter tuning. For each trained model, all the HI trajectories for the training set, i.e. all the 17 bearings, are computed and the mean monotonicity values for all these trajectories is used as the criterion for selecting the best hyperparameters values.

Here again, the fine-tuning for each strategy yields a similar architecture represented in Figure IV.19. The optimal hyperparameters values can be found in Table IV.7.

IV–3.2.5 Performance evaluation and discussion

(a) Setup #1

The first step in evaluating the proposed methodology on this dataset consists in testing the ability of learning a model when all the instances are taken for training. The results of this first experiment can be found in Figure IV.20 for the HI curves and on the first row of Table IV.8 for the

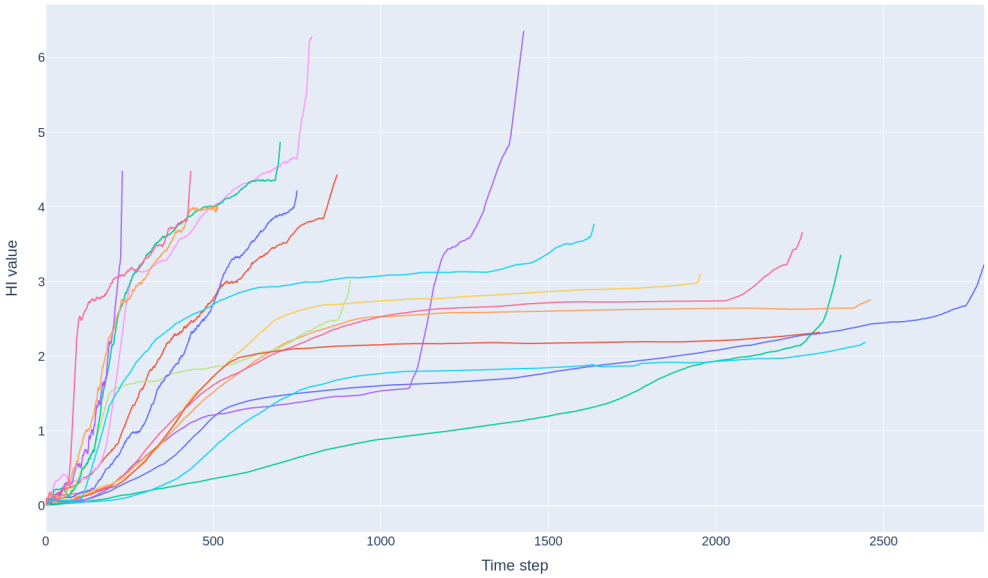
Hyperparameters			Trial values	Optimal values	
				Anchor at start	Double negative
Model	Core NN	e: SNN embedding size	10,20,50	50	
		n_{hd}: number of hidden dense layers	[1-3]	2	
		n_c: number of convolutional layer blocks	[1-3]	2	
Optim.	Optimiser	r: optimiser learning rate	0.01, 0.001, 0.0001	0.0001	
		n_B: batch size	16,32,64	16	
	samples selection	η: temporal neighbourhood for triplet selection	10,15,20	20	
	Anchor-at-start	λ: semi-hard margin	[1-5]	5	-
	Double-negative	m_{pen}: loss penalisation margin	0.1, 0.3, 0.5, 0.7, 1	-	0.1

Table IV.7: Hyperparameters of the method optimised for the bearing dataset.

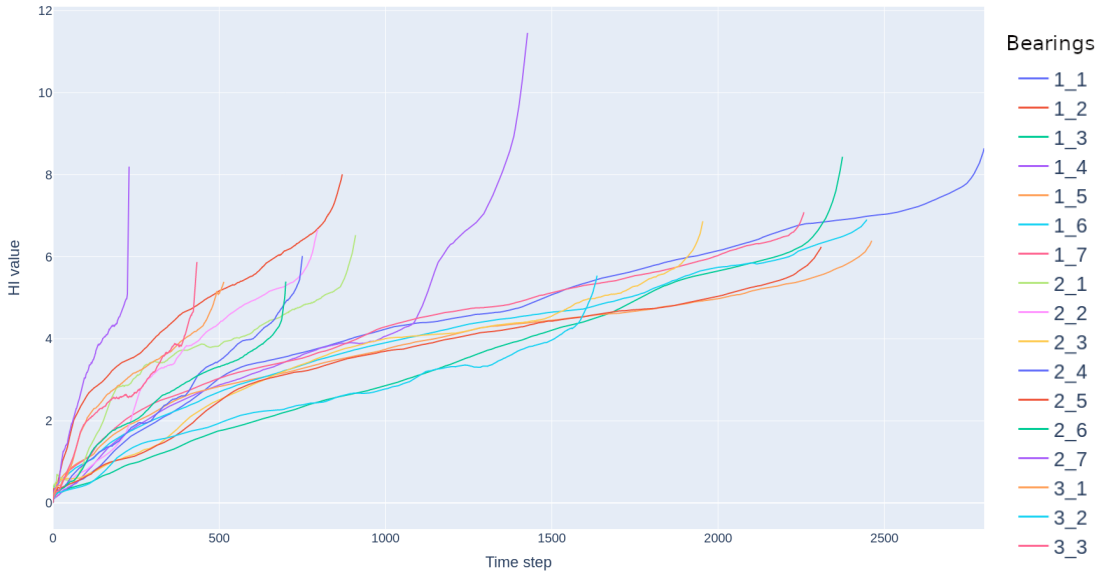
Set-up	Training data	Testing data	Strategy	*Mon*	*Rob*	*Pro*	*FC*
#1	All bearings	All bearings	Double negative	**1**	**0.007**	**0.897**	**0.876**
			Anchor at start	0.977	0.013	0.579	0.620
#2	LOO - All bearings	LOO - All bearings	Double negative	**0.527**	**0.051**	*	*
			Anchor at start	0.421	0.069	*	*
#3	Bearings 1_2, 1_4, 1_5, 1_6, 1_7, 2_1, 2_2, 2_3, 2_4, 2_5, 2_7, 3_1, 3_2	Bearings 1_1, 1_3, 2_6, 3_3	Double negative	**0.839**	**0.052**	**0.825**	**0.982**
			Anchor at start	0.828	0.063	0.650	0.809

Table IV.8: Performance indicators on bearing dataset test set (denoised HI).
* The criteria values for *Pro* and *FC* cannot be evaluated on test dataset holding a single bearing (LOO)

performance criteria averaged over all bearings. These results show that the proposed methodology is able to construct a general HI model on complex data such as bearing vibrations with perfectly monotonous HI curves, as evidenced by the value of monotonicity criterion for both strategies: 1 and 0.98. Another key finding from this experiment is that the **double-negative** strategy performs better on all the other criteria than the **anchor-at-start** strategy, with particularly satisfying results on the **failure consistency** criterion.



(a) Anchor at start



(b) Double negative Legend

Figure IV.20: HI curves obtained when training on all bearings.

(b) Setup #2

The second step of this experiment consists in investigating the ability of the model to generalise well on new data on which it did not learn. For this, the model is trained using the leave-one-out strategy on the entire dataset. The model is therefore trained 17 times, with a single bearing left out of the training set for each run. The evaluation criteria are then performed on the HI curves obtained for the leftover bearings for each run, and then averaged into a single value. The resulting HI performance criteria can be found in the second row of Table IV.8. A significant decrease in performance can be observed. This was expected given that, as opposed to the previous setup, the model was not trained on the bearings' data for which the performance criteria are calculated. The **double-negative** strategy reveals better performance once again, which is consistent with the

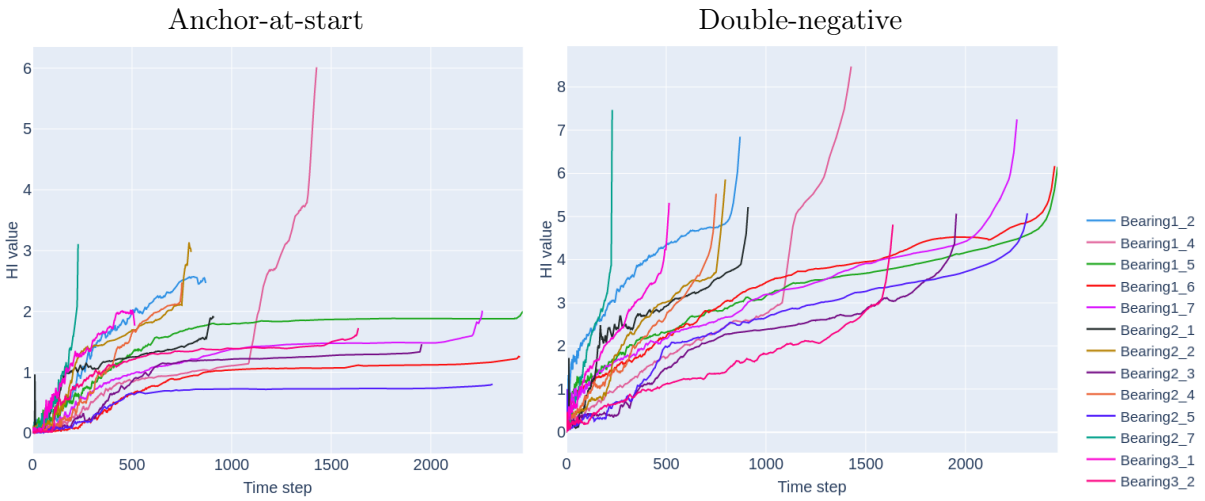
findings from Subsection IV–3.1.6. An additional thorough analysis of the performance criteria on individual bearings when left out of the training data reveals interesting results: there is a significant discrepancy in these criteria values among all bearings. This indicates that the vibration data for some bearings is very different from that of other bearings or at least under-represented in the given dataset, which inevitably leads to a generalisation problem and deteriorates the overall performance of the model. These differences in vibration data from bearings under similar operating conditions is reasonably well explainable by the complex internal dynamics of such systems and of the numerous possible modes of degradation it can experience. To illustrate this issue, five bearings are here identified as particularly problematic: 1_5, 1_7, 2_1, 2_3 and 2_5 as their monotonicity criterion falls under 0.3 in the **anchor-at-start** setting while the other bearings all reaches values above 0.5. By contrast, in the **double-negative** setting bearings 1_6, 2_6, 2_7 reach 0.7 and bearings 1_1 and 1_3 reach values even higher than 0.9. These differences in monotonicity performance criterion between the different bearings illustrate the problem faced with a dataset with a limited amount of data, here only 17 bearings. While for some bearings the generalisation is well performed by the proposed model, bearings that experienced degradation trajectories and possibly failure modes that differ to those represented in the training dataset can not be assigned a satisfactory HI.

(c) Setup #3

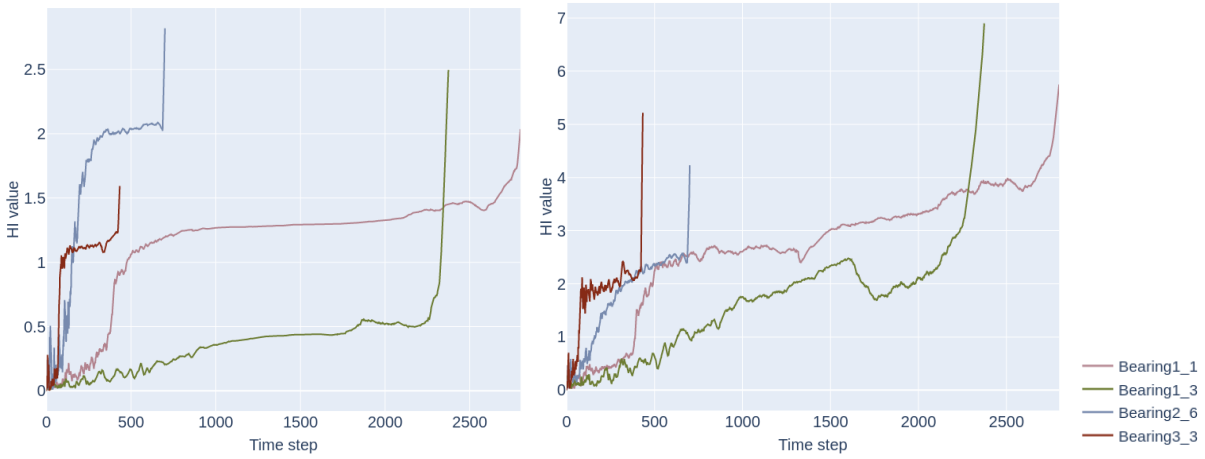
A third experiment is performed to validate the capacity of generalisation of the model on several instances of bearing that it was not trained on. To do so, we use the setting proposed in [216] where 4 bearings out of 17 (1_1, 1_3, 2_6 and 3_3) are kept out of the training set and used for testing the performance of the model. From setup #2 we can pinpoint that the generalisation is performed well for these 4 testing bearings suggested by [216]. It is therefore possible to get satisfactory results on this setting. The resulting HI curves are drawn in Figure IV.21, and the performance criteria obtained for this setup are available in the last row of Table IV.8. It shows very good results on the four criteria. More specifically, the **monotonicity** criterion reaches values higher than 0.8, which indicates that the model learns to construct monotonous HI and is able to generalise to unseen data, when representative of the training data. The **robustness** criterion is also satisfying, with residuals accounting for around 5% only of the HI values. Regarding the two other criteria, namely **prognosability** and **failure consistency**, the **double-negative** strategy shows better performances. On the former one, it gets a value higher than 0.8 indicating a strong correlation between the four HI curves of the testing bearings. On the latter one, it reaches a value close to 1, indicating a perfect failure consistency. For its part, the **anchor-at-start** strategy gets a value of **prognosability** significantly lower, which is consistent with the results obtained on the previous dataset and on the different settings of this dataset. For the **failure consistency** it reaches a value higher than 0.8 indicating a consistent behaviour in the end of life though significantly less than for the other strategy, which also confirms the results found on the previous dataset.

(d) Setup #4: Comparison with VAE-DC approach

Finally, a last experiment is conducted to compare the proposed approach, a SNN trained with a modified triplet loss, with a similar and recent work from the literature [150]. This specific work, which is mentioned at the end of Subsection III–1.2.1 for its particular relevance to the present study, is, to the author's knowledge, the only existing work that incorporates a monotonicity constraint into the HI construction model during the learning process. In contrast, other approaches rely on a



(a) Training set



(b) Testing set

Figure IV.21: HI curves on the bearing dataset, with 13 bearings for training and 4 for testing (**setup #3**).

preprocessing step, i.e. features filtering, as discussed in Subsection III–1.2.1. The authors of [150] propose to construct the HI with a variational auto-encoder (VAE) with a latent space of dimension one, i.e. the HI dimension. This approach can be classified as unsupervised fusion-based in the HI methods classification proposed in Chapter III. It can additionally be classified as a reconstruction self-supervised setting in the ML settings classification proposed in Chapter II. The authors of [150] incorporated a so-called degradation constraint (DC) that penalises non-monotonous HI evolution. To ensure a fair comparison, both the proposed approach and the method from [150] (VAE-DC) utilise the same neural network architecture for the encoder and the SNN clone, respectively. This involves preprocessing the data for the SNN similarly to the method proposed in [150], which relies on computing single-valued statistics on vibration samples, e.g. mean square, peak-to-peak or center frequency. The NN architecture used for both approaches is then adapted to fit this preprocessed data samples consisting of 15 real valued statistics for each of the two vibration measurements, i.e. a vector of 30 values. The SNN therefore then does not learn any more based on time-frequency images but rather simple vectors. The NN architecture used is a multi-layer perceptron with three layers

of 30, 15, and 7 neurons, using ReLU activations for the first two layers and a sigmoid activation for the last one. Because VAE-DC needs an encoder which outputs a single value, a last layer with a single neuron is added, the decoder is designed symmetrically. For further details on the precise design and training of VAE-DC, the reader may refer to [150]. Both model are trained with **setup #3**. The testing HI curves resulting from this experiment are drawn on Figure IV.22 , the indicator values for the testing set are summarised in Table IV.9 . The indicators suggest that both strategies of SNN presented in the present work outperform the method proposed in [150]. More precisely, the **monotonicity** is improved with SNN and even largely improved with the **anchor-at-start** strategy, which reaches a value of **monotonicity** close to that obtained with the CWT preprocessing. The failure consistency however is slightly better with the VAE-DC approach, even if the SNN obtains satisfactory values above 0.8. Overall, the performance results indicate that the SNN strategy leads to HI curves that are more monotonous than with the VAE-DC strategy. Additionally, combining the preprocessing of the vibration signals with CWT and using CNN to build the SNN's clone lead to improved results, see Table IV.8 setup #3, compared with the preprocessing proposed in [150] combined with dense NN.

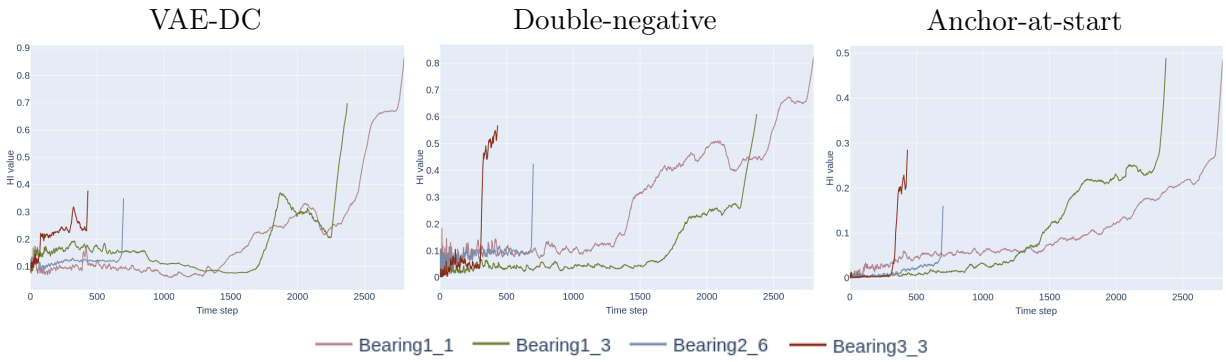


Figure IV.22: HI curves of testing set resulting from VAE-DC method and proposed SNN method for the two strategies: **anchor-at-start** and **anchor-at-start**

Model	*Mon*	*Rob*	*Pro*	*FC*
VAE-DC	0.314	**0.019**	0.329	**0.912**
SNN Double negative	0.4425	0.102	0.376	0.84
SNN Anchor at start	**0.739**	0.107	**0.765**	0.821

Table IV.9: Performance indicators on bearing test dataset (denoised HI) resulting from VAE-DC method and proposed SNN method for the two strategies: **anchor-at-start** and anchor-at-start

IV–4 Conclusion

In this chapter, a similarity-based HI construction method is proposed to enforce monotonicity of the resulting HI. It is built upon a self-supervised contrastive learning setting with a SNN as the representation learning model, two distinct training samples selection and / or, a constraint incorporated directly in the contrastive loss function. No additional particular assumption on the evolution of the HI is therefore needed, which is a significant novel idea from the literature and make the proposed method very generic and adaptable to any physical system.

The turbofan experiment in Subsection IV–3.1 shows the efficiency of this method for enforcing the monotonicity on a dataset with relatively high dimension in the MTS and also a high number of system's instances. A clear correlation between the improvement of the loss of the model and the resulting monotonicity of the HI has been established for two distinct strategies of monotonicity constraints.

The bearing experiment in Subsection IV–3.2 shows the efficiency of the proposed model on reaching its aim for datasets with a reduced number of system's instances, where the MTS are sampled at high-frequency, and whose system degradation dynamic is known to be complex and challenging to learn. The proposed method seems to perform better than the previous work of [150] on the matter that attempts to reach the same purpose.

The combination of these two experiments proves the high adaptability of the proposed methodology for different contexts of data, low or high number of dimensions, low or high frequency sampling, low or high number of system instances, and different MTS preprocessing techniques.

V

Usages of HI for prognosis and health management tasks

In the previous chapter, a new generic methodology has been proposed for learning a HI model. This latter is able to assess the HI value at any instant t of any instance of the system based on the latter own historical measurements up to t. This is done by learning from historical measurements of run-to-failure instances of the system. As a result, a set of complete HI trajectories of run-to-failure instances can be obtained: $\mathcal{HI} = \{HI^{(k)}(t)\}_{k\in\mathcal{K},\, t\in\mathcal{T}_k(T^{(k)})}$. These HI trajectories can in themselves be of particular interest to manufacturers, maintenance planners and users of the systems under study. However, in this chapter, we investigate the potential of exploiting these HI trajectories, but also, in the case of the approach proposed in Chapter IV, of using the learned latent space, to extract even more useful knowledge for the prognostic and health management of these systems.

First, this chapter proposes to study the application of HI trajectories for identifying failure modes in a first section. If RUL prognostics is the preferred use of HI trajectories, we also consider that these HI trajectories also offer insights into the degradation specificities of individual instances of each system. To divide these trajectories into several clusters, an univariate time series clustering technique is here applied to the HI trajectories obtained in Chapter IV. Comparing the obtained clusters with the actual true distribution of failure modes in various system instances help us determining whether these trajectories are useful for identifying failure modes.

Then, as discussed in Chapter III, a natural application of HI learning is predicting the remaining useful life (RUL). This may involve combining a HI forecasting method with a failure threshold estimation technique. Taking inspirations from the work of [21] and [78] we here propose a general, rather than individual (cf. Subsection III–2.1), forecasting model coupling a particle filter and an ensemble of neural network.

Finally, a second use of the method proposed in Chapter IV for RUL prediction is also explored. This time, instead of using the HI trajectories, we only employ the latent space created by the siamese neural network. Considering the RUL prediction in the context of contrastive self-supervised learning (cf. Subsection II–2.2), the approach presented in Chapter IV can be used as a pretext task that generates a latent space where the measured samples are projected and represented in a manner customised for further RUL prognostics. This second approach explores the possibility of using the projection of measured samples in the latent space as inputs of a direct RUL

prognosis that links any of its input to a RUL estimation. This type of approach has, to the authors knowledge, never been proposed and is an interesting direction of future research.

V–1 Failure mode identification

For a given system or device, a HI curve expresses how the system or device degrades over time and, to some extent, whether or not it is likely to fail in the future. Hence, we believe that failure modes can be identified by analysing HI curves. We therefore propose to predict the driving failure mode of a system by resorting to a clustering approach. To the best of the authors' knowledge, this has never been done in the past. One should note that the purpose is not to provide new clustering tools as it relies on existing clustering strategies and distance metrics in order to cluster failure modes from HI curves. The proposed approach then assumes a set of HI curves obtained from several instances of a given system $\mathcal{HI} = \{HI^{(k)}(t)\}_{k\in\mathcal{K},\, t\in\mathcal{T}_k(T^{(k)})}$.

V–1.1 Time series clustering

Many methods already exist for clustering univariate time series and have been thoroughly studied in literature. According to the review and survey in [2, 206] respectively, the most representative techniques are partition clustering (e.g. k-means or k-medoids), model-based clustering (e.g. self-organising maps) and hierarchical clustering (e.g. agglomerative clustering). Both works mention the difficulty of partition and model-based clustering when dealing with time series of unequal lengths. In our application case, the time series to be clustered do not have equal lengths, therefore these two approaches are discarded as potential candidates. In contrast, both papers point out the ability of agglomerative clustering for dealing with time series of unequal lengths if an adequate distance metric is chosen. Elastic distance measure in general, and dynamic time warping (DTW) in particular are especially mentioned in such cases. This literature analysis thus led here to the choice of agglomerative clustering with DTW distance for HI curves of unequal lengths.

V–1.2 Dynamic time warping

DTW is an algorithm that can be used to measure the similarity between curves, trajectories, sequences of values or more generally time series. Its strength resides in its ability to compare such elements even if they do not have matching lengths, numbers of points or sampling frequencies. As opposed to classical distance measures that need matching length, DTW does not compare elements of the sequence that have necessarily the same position in the sequence. DTW indeed, allow for one element of the the first sequence to be compared against one or multiple elements of the second sequence. This mapping between sequences elements is the core of DTW and is visually illustrated in Figure V.1. For such a mapping to be relevant for DTW it should satisfy some properties defined as follow:

- Every element of one of the two sequences must be mapped to at least one element of the other one.

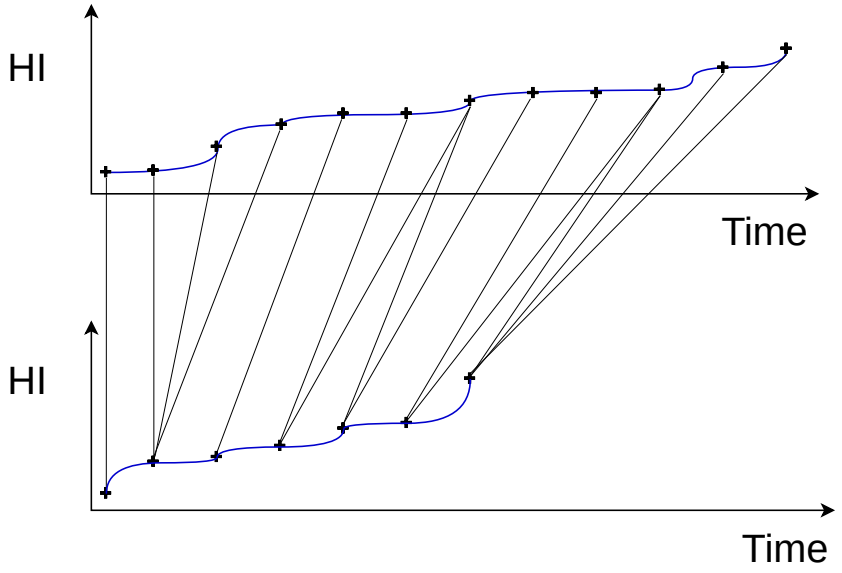


Figure V.1: Example of a valid mapping for DTW for two sequences of different length, sampling and number of points.

- First and last elements of both sequences must respectively be mapped together. They can still be mapped to other elements as well.
- The mapping between two elements must respect the order of the sequence. If a and b are two elements from the first sequence such that $a > b$, and c and d are two elements from the second sequence such that $c < d$, then it is not possible to map a with c and b with d.

Many mappings can of course be valid, using only these rules. DTW aims at finding the mapping that minimise the obtained distance between the two sequences. This distance is simply the Euclidean distance considering the found mapping instead of a classical mapping depending on the strict position in the sequence. It is worth mentioning that the DTW used for the HI trajectories in this work is applied only on one dimension. The time dimension is not considered as a component of the HI trajectories elements, it is rather only used for determining the position of each HI value in the sequence. This choice is motivated by the removal of the influence of the length of HI trajectories for their clustering.

V–1.3 Agglomerative clustering

Agglomerative clustering is based on two elements: a metric (here DTW) defining the distance between the data samples to be clustered and a linkage criterion, i.e. a way to compute the distance between two clusters based on the distance between their respective samples. In agglomerative clustering, each data sample starts in its own cluster. Then, until a criterion for ending the clustering is reached, the two closest clusters, according to the linkage criterion, are merged together. Usually the criterion for ending the clustering is that all the observations are merged in a single cluster. This enables a complete representation of the different sub-clusters hierarchy, called a dendogram. In this dendogram each value of the linkage criterion is then associated to a clustering level, the lower the linkage value, the more clusters there are. An example of a dendogram resulting from

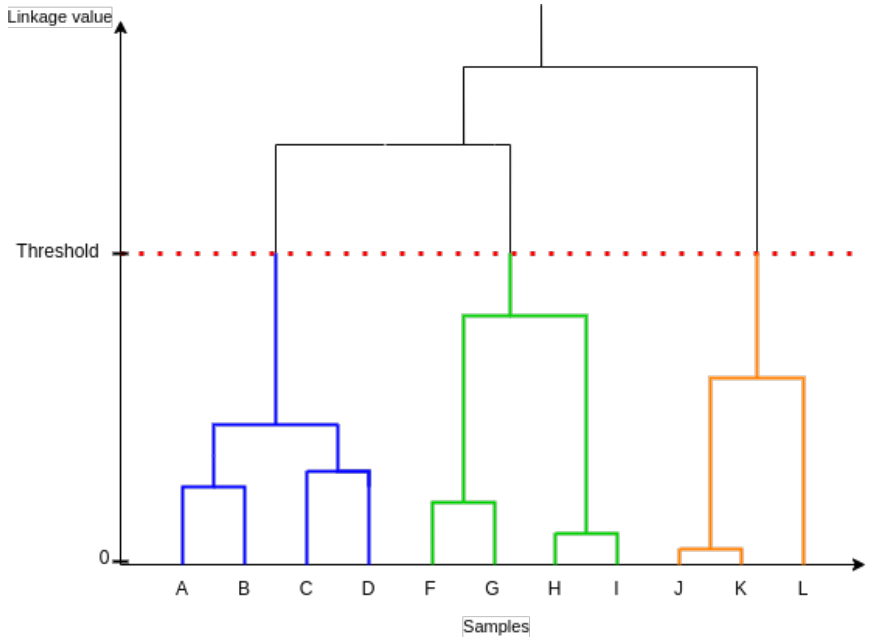


Figure V.2: Dendogram example: here the threshold value sets the clustering with 3 clusters, (A,B,C,D), (F,G,H,I) and (J,K,L)

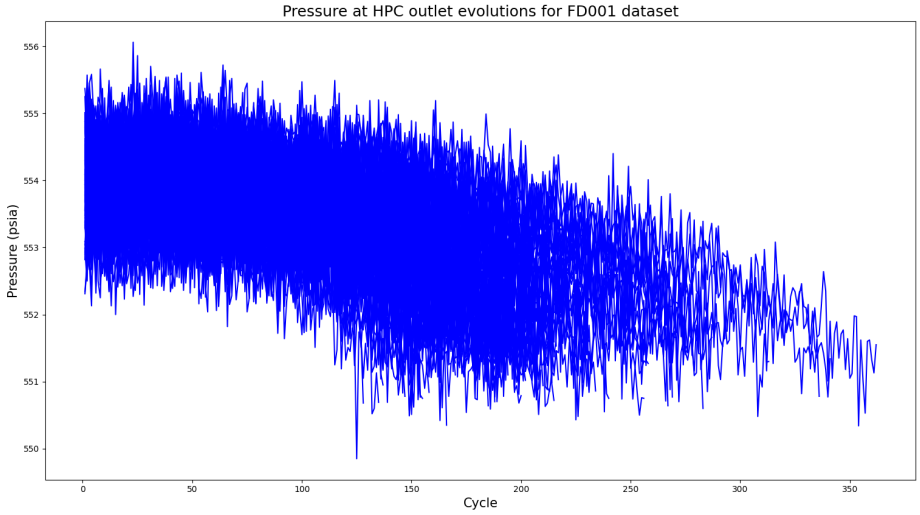
agglomerative clustering is given in Figure V.2.

The existing linkage criterion that can be used in agglomerative clustering based on DTW distance are the followings: single, average, complete and Ward. Single linkage defines the distance between two clusters as the minimum distance of two elements taken in each of the two clusters. Complete and average linkage are similar to single linkage but retains the maximum and average distance, respectively. The Ward linkage defines the distance between two clusters as the additional total within-cluster distances variance obtained if the two clusters are merged.

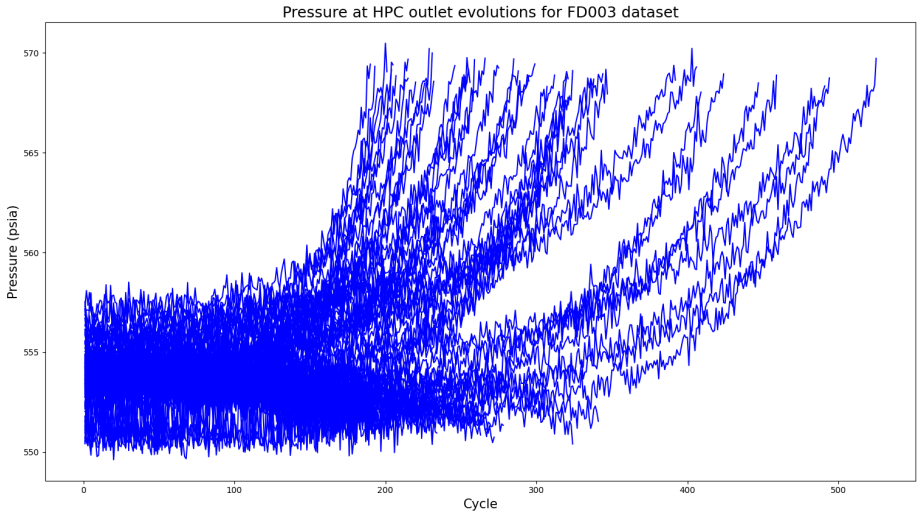
V–1.4 Experiments

V–1.4.1 Application to turbofan dataset

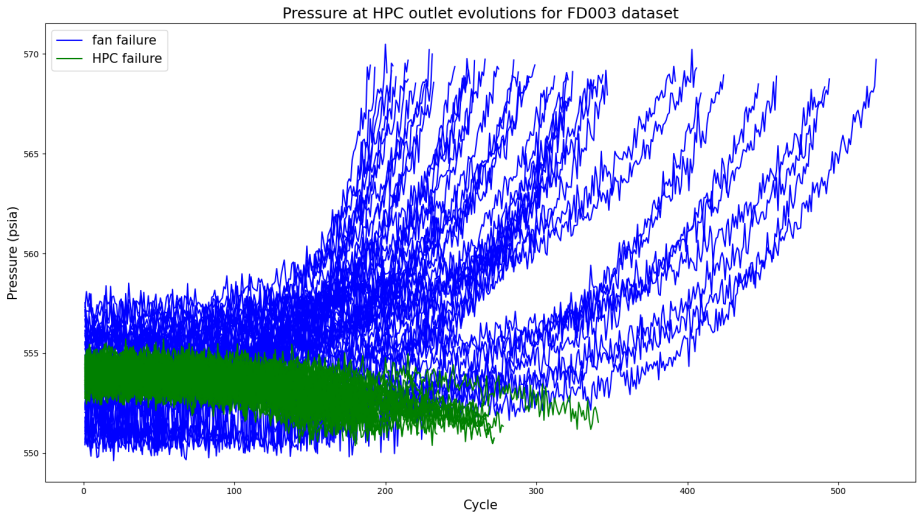
For assessing the use of HI trajectories to cluster different instance of a system into different cluster representative of their failure modes, we decided to use the Turbofan dataset already presented in this work. This dataset is particularly interesting for such an investigation because we already know the number of failure modes present in the available run-to-failures. However, there is no direct information in the dataset on which run-to-failure (RtF) experienced which failure mode. This information is extremely valuable for assessing the ability of the clustering approach to correctly distinguish between different failure modes. A first investigation was therefore to find this information in the available data.



(a) Turbofan HPC pressure evolution for FD001 dataset.



(b) Turbofan HPC pressure evolution for FD003 dataset.



(c) Turbofan HPC pressure evolution for FD003 dataset discriminated by failure mode.

(a) Prior failure mode identification in turbofan dataset

In the original paper presenting the turbofan datasets [166], the authors present the four created datasets. It is mentioned that datasets FD003 and FD004 contain turbofan run-to-failure data with two possible modes of degradation: high-pressure compressor (HPC) or fan failure, while the datasets FD001 and FD002 only contains HPC failure. Unfortunately, the paper does not provide the failure labels for each turbofan's run-to-failure in FD003 and FD004. To extract this information we have analysed the given signals in the dataset related to either the HPC or the fan: temperature at fan inlet, temperature at HPC outlet, pressure at fan inlet, pressure at HPC outlet and fan speed. It turned out that the unique measurement of pressure at HPC outlet is sufficient to discriminate both failure modes. Figure V.3a shows the different curves of HPC pressure evolving with time for the FD001 dataset. It is clear that all turbofans follow the same trajectories on this measurement, i.e. a slight decrease over time. In Figure V.3b however, some turbofans from FD003 follow a different trajectory, i.e., exponential growth while the others follow the same trajectory as in FD001. We thus discriminated the HPC failures and fan failures via the trajectories of the HPC pressure along the life of the turbofans, as shown in Figure V.3c. If the trajectory exhibits a slight decrease then the turbofan experiences HPC failure, if it has an exponential growth then it experiences fan failure.
Thanks to this analysis we were able to label each RtF with its supposedly corresponding failure mode.

(b) Results

The hierarchical agglomerative clustering with DTW presented in this section is then performed on the HI curves of the training dataset obtained in Chapter IV with the **double-negative** strategy. After a comparison of possible linkage criterion on the studied application case, it is found that the clusters hierarchy found by the Ward criterion is the most balanced. Ward linkage is therefore used for analysing the results of the experiment. The resulting dendogram is represented in Figure V.4, while the clustered trajectories are drawn in Figure V.5

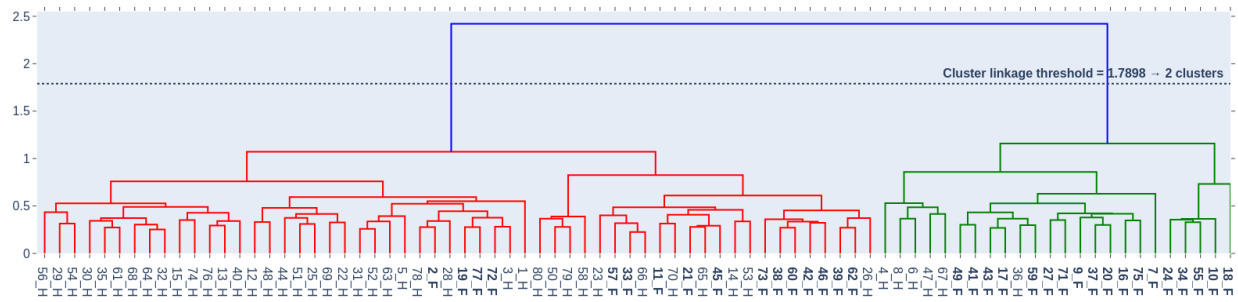


Figure V.4: Dendogram of hierarchical clustering for turbofan HI curves of training set
On the x-axis, the turbofan units with a Fan failure have a bolded label "**F**" when the HPC one have a normal label "H".

This dendogram shows that two very distinct clusters can be obtained by setting the linkage threshold accordingly. It is worth mentioning that the linkage value range (Y-axis), in which a selected threshold value results in two clusters, is large, which emphasises the clear separation

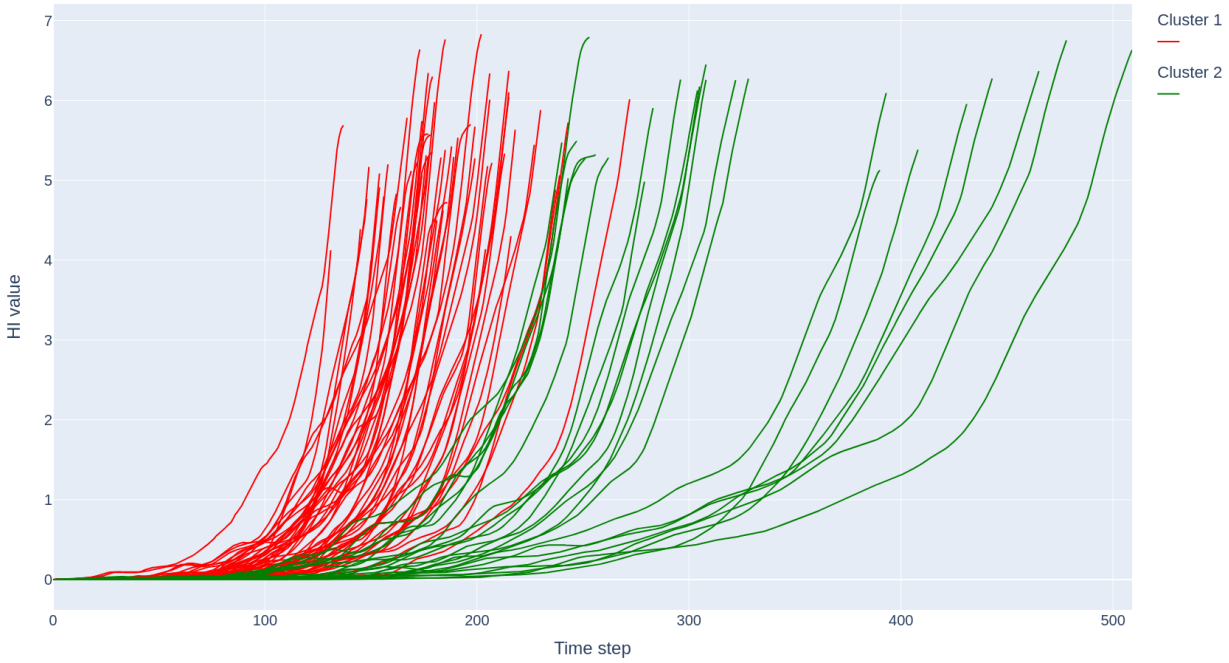


Figure V.5: Clustered HI trajectories for the turbofan dataset (**double-negative** strategy)

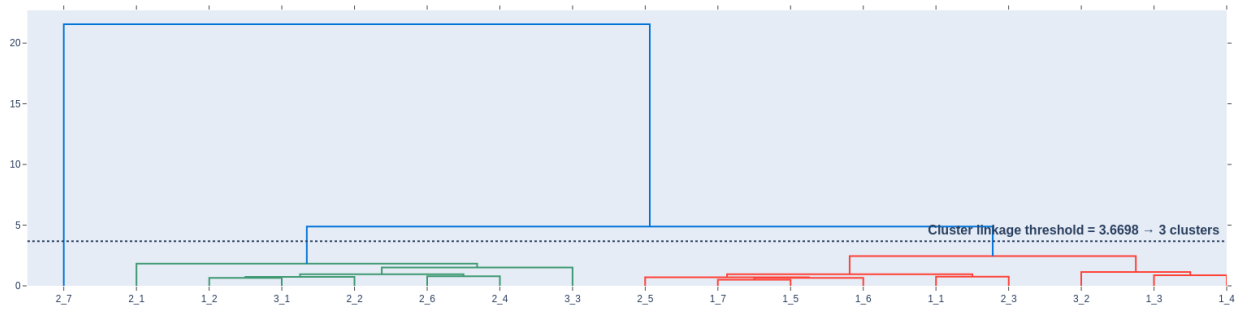
between the two clusters. The separation between Fan and HPC failures is clear, although not perfect. The "fan cluster" (in green in Figure V.4) is composed of 18 fan failures denoted by "F", and 6 HPC failures denoted by "H", while the "HPC cluster" (in red in Figure V.4) is composed of 16 fan failures and 40 HPC failures. These promising results indicate that it may be possible to use the constructed HI curves to identify different degradation dynamics representative of different failure modes. It is important to remind that the proposed HI construction method is not optimised for this task of failure mode clustering. It nevertheless demonstrates the ability to produce HI curves with shapes somehow depending on their failure mode. We believe that integrating failure mode identification in RUL prognosis by the mean of HI curve clustering tasks offers the particular benefits of gaining more information on the studied system. This also might to a certain extend, in the end, improve the following steps needed for RUL prognosis, i.e. failure threshold definition and HI curve forecasting.

V–1.4.2 Application to bearing dataset

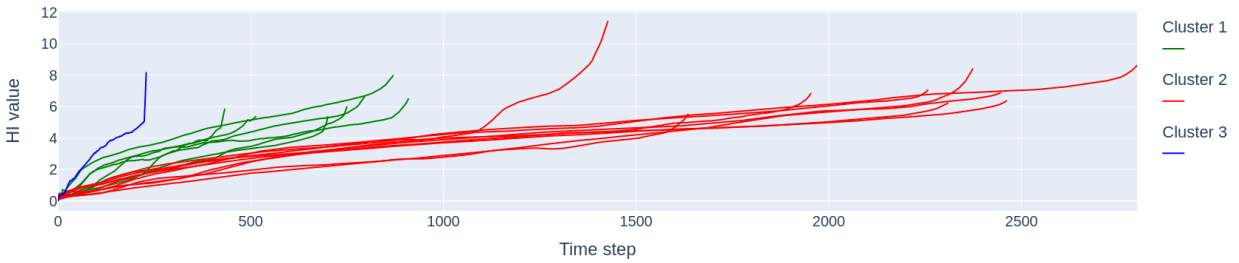
To explore the potential differences in degradation on the bearing dataset, the HI trajectory based clustering is also applied on this dataset. However, because no information is available on the failure modes experiences by each bearing, it is impossible to conclude on the efficiency of the method to discriminate between different failure modes for this dataset. This experiment is then more exploratory, to identify potential clusters of degradation trajectories in the bearing dataset. The **setup #1** from Subsection IV–3.2 of HI training on bearing was used. That is to say that all bearings are used as training.

(a) Results

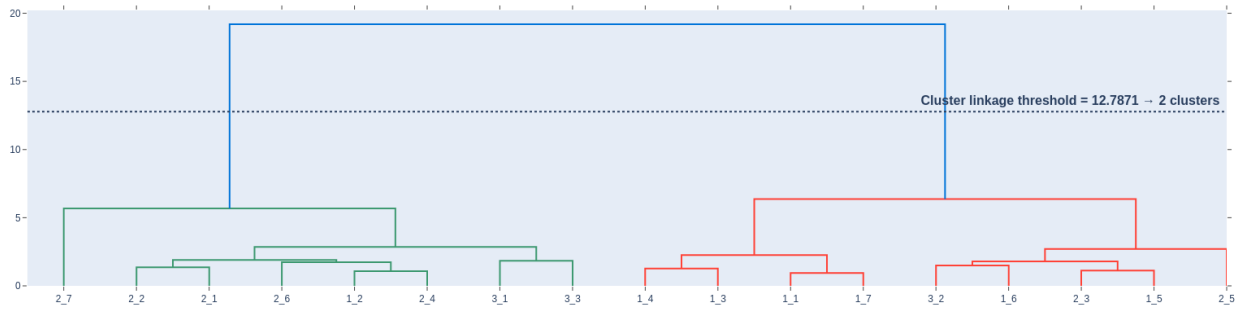
The resulting HI trajectories could then be used for our proposed HI based clustering, the results of which are shown on Figure V.6. The conclusion that can be reached when looking at the clustering results is that HI curves with short lives are clearly separated from HI curves with long lives in two distinct clusters. This could indicate that the degradations observed on the short-life bearing might be of different nature than the ones on the long-life bearings. This cluster separation could also be linked to the rate of increase of the trajectories. Indeed, the HI trajectories of the green cluster in Figure V.6 have significantly higher gradients compared with the one of the red cluster, especially at the beginning of their lives. This could also indicate a difference in failure modes, with one failure mode more likely to happen in the beginning of life, and another one more related to a slow degradation. However, we should once again note that the failure modes for each bearing are unknown, and we cannot conclude here on the ability of clustering HI trajectories to identify classes of failure modes.



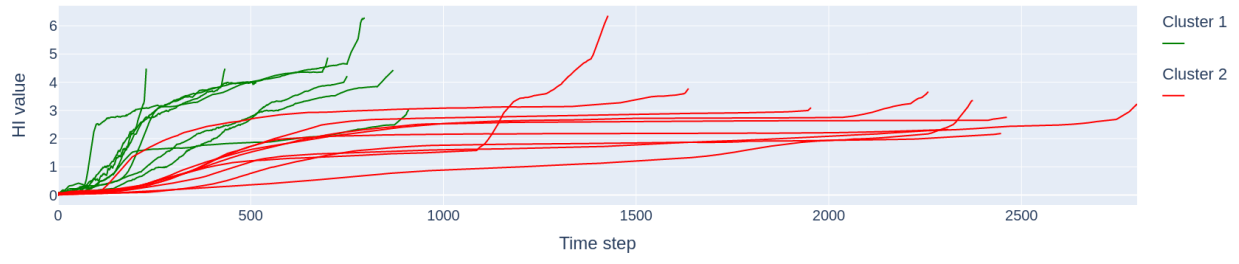
(a) Dendogram of clustering for the **double-negative** strategy



(b) Clustered HI trajectories for the **double-negative** strategy



(c) Dendogram of clustering for the **double-negative** strategy



(d) Clustered HI trajectories for the **anchor-at-start** strategy

Figure V.6: Clustering results of HI trajectories on the bearing dataset

V–2 RUL Prognosis

The second investigation of HI trajectories usage is, as mentionned before, focused on RUL prognostics, and is outlined in this section. It is separated in two different strategies, the first one using the HI trajectories in a particle filter approach and a second one using the latent space where the HI trajectories are computed.

Choice of notations

In this section and for the sake of simplicity, the notation $x_{0:t}$ is used to define the evolution of any value x from an initial time 0 to the current time t.
The notations of the type x_0, x_t or x_{t-1} only denotes for the value of x at time 0, t or $t-1$ respectively.

V–2.1 HI forecasting combined with failure threshold estimation: a particle-filter based approach

As mentioned in Subsection III–2.1, the HI-based RUL prognostic, is typically composed of a HI forecasting method combined with a failure threshold estimation one. It has been developed in this same section that a particular method combining particle filters and neural networks recently emerged in the literature for the task of HI forecasting [20]. In this section we propose some improvements to this existing method. But before diving into the details of these proposed improvement the next subsection introduce the method of particle filters.

V–2.1.1 Particle filter

Particle filter (PF), also called sequential Monte Carlo [45], is an algorithm to sequentially estimate the posterior probability density function (PDF) of a hidden state x_t given a series of noisy observations $z_{0:t-1}$, t where t denotes the current time step. The state-space evolution is described as a discrete hidden Markov model defined by:

$$x_t = f(x_{t-1}, \omega_{t-1}) \tag{V.1}$$

$$z_t = g(x_t, \eta_t) \tag{V.2}$$

where f in Eq. (V.1) describes the state equation and g in Eq. (V.2) the measurement equation, and where ω_t and η_t are the state and measurement noises, respectively and both considered normally distributed with zero mean. To recursively estimate the posterior PDF $p(x_t|z_{0:t})$, the algorithm uses the prediction-update recurrence. First, a prior distribution $p(x_t|z_{0:t-1})$ is computed using the measurement available until time step $(t-1)$. The distribution is built by generating N_p state trajectories, also called particles, based on the state model defined in Eq. (V.1). The second step consists in an update of the distribution. To do so, an importance weight is assigned to each simulated particle $q = 1, \ldots, N_p$:

$$w_t^{(q)} = w_{t-1}^{(q)} p(z_t|x_t^{(q)}); \tag{V.3}$$

in which $p(z_t|x_t^{(q)})$ is the likelihood $\mathcal{L}_t^{(q)}$ of the particle q given the true observations z_t. The formula for this likelihood typically depends on how the function g is defined. In this formulation of PF it is clear that the Markov hypothesis is considered true, i.e. the observation at any time t can be estimated using the state at the same time t which in turn can be estimated using the state at the previous time $t-1$.

In its original form, PF suffers from the degeneracy problem [7]. All particles but one tend to have an importance weight close to zero, hence the entire distribution of particles collapses to one single particle. A common way of tackling this issue is to resort to the sampling importance resampling (SIR) scheme [83]: The importance weights are normalised as follows:

$$\hat{w}_t^{(q)} = \frac{w_t^{(q)}}{\sum_{j=1}^{N_p} w_t^{(j)}} \tag{V.4}$$

and the trajectories $x_t^{(q)}$ are resampled on the basis of the normalised importance weights, i.e. by sampling with replacement. This means that the trajectories with high importance weights are more likely to be resampled, thus avoiding the degeneracy problem, introducing on the other hand a loss of diversity on the particles [45].

V–2.1.2 Surrogate-based PF

The classical PF methodology relies on the availability of the state model expressed in Eq. (V.1) and on the measurement model defined at Eq. (V.2). However, in the PHM context, the use of a VHI with no physical meaning, and thus no evolution model, renders the state and measurement equations unknown. To solve this issue, a common method consists in substituting the measurement equation $g(x_t, \eta_t)$ with a parameterised surrogate model $\widetilde{g}(t; x_t) = \widetilde{g}_t(x_t)$ to which the white noise η_t is added. This surrogate function is applied on t. The hidden states x_t are then considered as the parameters of this surrogate model $\widetilde{g}$ to which the white noise ω_{t-1} is also simply added at each time steps, thus defining the function f of Eq. (V.1), see eqs. (V.5) and (V.6) updated accordingly.

The PF state-space formulation is now rewritten as:

$$x_t = x_{t-1} + \omega_{t-1} \tag{V.5}$$

$$\widetilde{z}_t = \widetilde{g}_t(x_t) + \eta_t \tag{V.6}$$

in which x_t is now a state vector containing the parameters of the surrogate model $\widetilde{g}$ at time step t.

In the surrogate-based PF used in this work, the likelihood of each particle q at each time step t is calculated as follows:

$$\begin{aligned} \mathcal{L}_t^{(q)} = p(z_{0:t}|x_t^{(q)}) = ((2\pi)^{t+1}|\Sigma_\eta|)^{-0.5} \\ \exp\left\{-\frac{1}{2}\left(z_{0:t} - \widetilde{g}_{0:t}(x_t^{(q)})\right)^T \Sigma_\eta^{-1}\left(z_{0:t} - \widetilde{g}_{0:t}(x_t^{(q)})\right)\right\} \end{aligned} \tag{V.7}$$

Here, Σ_η and Σ_η^{-1} denote for the covariance matrix and its inverse of the observation noises $\eta_{0:t}$ respectively filled with σ_η^2 and $\frac{1}{\sigma_\eta^2}$ on their descending diagonals and zeros elsewhere. $z_{0:t}$ and $\widetilde{g}_{0:t}(x_t^{(q)})$ are $(t+1)$-sized vectors gathering the values until time t respectively of the observations

and the trajectory defined for $\widetilde{g}_{0:t}$ by the state of the particle q at time t.

$$\widetilde{g}_{0:t}(x_t^{(q)}) = [\widetilde{g}(0; x_t^{(q)}), \ldots, \widetilde{g}(t; x_t^{(q)})] \tag{V.8}$$

It is worth noticing that this two terms $z_{0:t}$ and $\widetilde{g}_{0:t}(x_t^{(q)})$ in Eq. (V.7) indicate that $\mathcal{L}_t^{(q)}$ is the likelihood of particle q at time t given the entire set of available measurements until t. This is in contrast with the classical PF in which the likelihood is usually computed just based on the last observation, exploiting the Markov hypothesis, see Eq. (V.3).

The evolution of the model parameters in Eq. (V.5) is obtained by adding a random perturbation as a Gaussian noise ω_{t-1} with zero mean to the previous state vector x_{t-1}, which enables the exploration of the parameterised function family of $\widetilde{g}$. This exploration is thus modulated by $\Sigma_{\omega,t}$ the covariance matrix of the state noise. Because the surrogate function $\widetilde{g}$ typically depends on a certain number of parameters denoted here n_θ, the state x_t and noise ω_{t-1} are thus vectors of size n_θ. For the sake of simplicity it is herein considered that the noise vector ω_t is independent through its n_θ components and therefore $\omega_t \sim \mathcal{N}(0, \Sigma_\omega^{(t)})$ where:

$$\Sigma_\omega^{(t)} = \begin{bmatrix} \sigma_{\omega,t}^2[1] & & \\ & \ddots & \\ & & \sigma_{\omega,t}^2[n_\theta] \end{bmatrix} \tag{V.9}$$

The choice of variances applied on each state dimension $\sigma_{\omega,t}^2[d]\ \forall d \in [1, n_\theta]$ is of prime importance: too low values would not guarantee a large enough exploration of the state space, while too large values would result in the algorithm instability. Moreover, each state parameter do not necessarily have the same sensitivity regarding perturbations which would hence necessitate that $\sigma_{\omega,t}[d] \neq \sigma_{\omega,t}[d'],\ \forall d \neq d'$. This issue of parameter-dependent noise variance is typically ignored in the formulation proposed in [20], a solution to overcome this limitation inspired from the theoretical development on particle filter optimal jittering in [174] is proposed here, see paragraph (d) and is one contribution of this chapter.

V–2.1.3 Algorithm Details

(a) MLP surrogates

A combination of PF and NN has been firstly proposed by [51] and, as mentionned previously, has later been investigated for HI forecasting with the aim of RUL prognostic of Li-Ion batteries in [20]. In this work, inspired by this lastly mentioned reference, we decide to resort to a simple MLP architecture composed by four layers (one input, two hidden layers and one output layer), with three neurons per hidden layer. The input is a single value, i.e. the time step t, and the output is composed of a single neuron returning the predicted HI value. The MLP is then composed of 15 weights and 7 biases, for a total of 22 parameters to completely describe the network represented in Figure V.7. The input-output relationship is described as:

$$\widetilde{HI}_t = z_t = \widetilde{g}_t(x_t) = \widetilde{g}_t(w_t, b_t) = h_O(\sum_{j=1}^{3} w_t^{(j)} H_2^{(j)}(t) + b_t^{(O)}) \tag{V.10}$$

$$H_2^{(j)}(t) = h_2(\sum_{i=1}^{3} w_t^{(i,j)} H_1^{(i)}(t) + b_t^{(j)}) \tag{V.11}$$

$$H_1^{(i)}(t) = h_1(w_t^{(i)} t + b_t^{(i)}) \tag{V.12}$$

where $x_t = \{w_t; b_t\}$ represent the MLP weights and biases and thus the states in the PF state-space equations in eqs. (V.5) and (V.6), we thus have $n_\theta = |\{w_t, b_t\}|$. i and j respectively refer to the neuron number of the first and second hidden layers, while h_O, h_2, h_1 refer to the activation functions of the output, second and first hidden layers, respectively. In this work, the activation functions $h_O(\cdot)$, $h_2(\cdot)$ and $h_1(\cdot)$ are respectively set as linear, exponential and scale exponential linear unit (SELU) functions. The choice of $h_2(\cdot)$ and $h_O(\cdot)$ is motivated to fit a sum of exponential, which is well fitted to an exponential-like degradation. The first hidden layer activation function choice is for previously adding some non-linearity, expanding the possibilities of the defined parameterised function family, and therefore improving its adaptability to degradation trajectories containing exponential growths. It is also advised in the literature on NN to use pseudo-linear unit activation functions e.g. rectified linear unit (ReLU), SELU or ELU.
The choice of such a simple architecture is motivated by the low computational efficiency of PF, whose task is to recursively update the MLP parameters x_t. Increasing the network complexity could on one side improve the approximation capability of the surrogate $\widetilde{g}$, but on the other side would drastically increase the number of parameters and the computational burden of the entire PF algorithm. Additionally, the more parameters there are in the surrogate function, the more the size of the states and the more particles are needed to approximate the multivariate distribution of these latter.

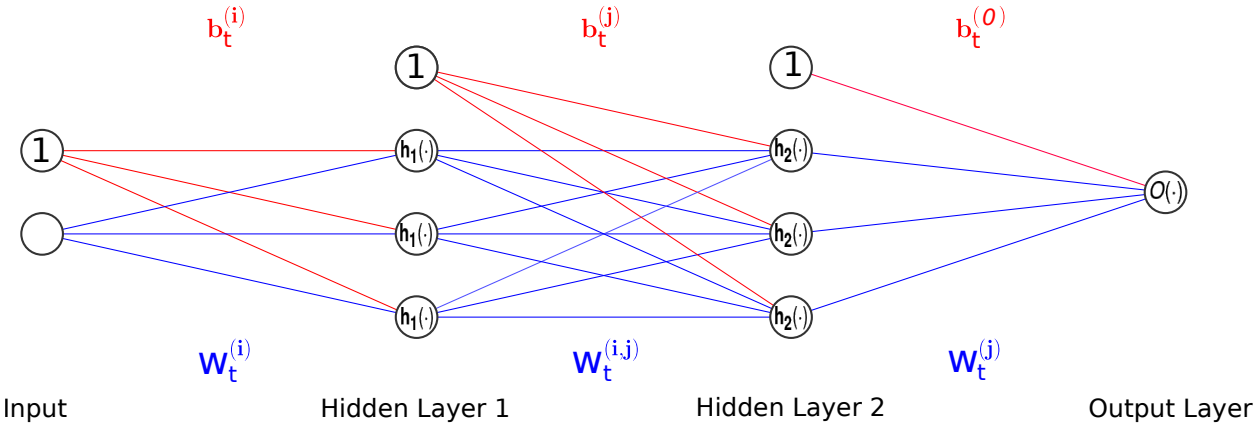


Figure V.7: Proposed MLP architecture

(b) Particle initialisation and reintroduction

The initialisation of the PF with plausible particles plays a crucial role in the algorithm performances. Randomly initialising the particles, i.e. the parameters x_0 in Eq. (V.6), would lead to poor results, especially with $\widetilde{g}$ lying in a large parameterised function family as defined in the previous section. To do so, it is here proposed to initialise the particles with parameters x_0 estimated when fitting $\widetilde{g}$ on HI trajectories selected to compose the training set. Once all the training trajectories are fitted with their optimal parameters, a particle q is initialised by randomly taking the parameters of one training trajectory fit. Additionally, to avoid a decrease in the diversity of the particles along the PF cycles, initial particles corresponding to initial training trajectories fit are randomly reintroduced at each re-sampling stage. The percentage of particle reintroduced instead of being

re-sampled is here arbitrarily fixed at 10% but could be considered as an hyperparameter of the method to be fine-tuned. For learning $\widetilde{g}$ with MLP surrogate models, a classical neural network training is achieved to minimise the mean square error on the training curves.

(c) Particle likelihoods variance

To determine the likelihood variance σ_η^2 used for the particle weight estimation performed in Eq. (V.7), it is first needed to characterise the noise η defined in Eq. (V.6). This noise is estimated off-line by the difference between known HI trajectories, i.e. the training set, and their respective fits $\widetilde{g}_t(x_0)$. Fortunately, these best fit parameters for each training trajectories are already computed for the initialisation of particles, so this step does not require any additional computational burden. The mean $\widehat{\mu}_\eta$ and variance $\widehat{\sigma_\eta^2}$ of η are then estimated over all time steps of the training trajectories. It was here found that the mean was close to zero for all MLPs, which justifies the hypothesis of zero mean gaussian noise η. In the PF, the value used for σ_η^2 should not be lower than $\widehat{\sigma_\eta^2}$. Otherwise, a too low weight would be assigned even to the particles that best fit the HI trajectory, resulting in minimal distinction between the weights of "good" and "poor" particles. In this study, several values for the parameter σ_η^2 were subsequently explored based on the estimated noise variance $\widehat{\sigma_\eta^2}$. More specifically, the values $[\widehat{\sigma_\eta^2}, 2\widehat{\sigma_\eta^2}, 5\widehat{\sigma_\eta^2}, 10\widehat{\sigma_\eta^2}, 20\widehat{\sigma_\eta^2}]$ were tested. Here, for the proposed MLP, $\widehat{\sigma_\eta^2}$ was estimated to be approximately 0.015, and the best results of PF were achieved with $5\widehat{\sigma_\eta^2} = 0.075$.

(d) Jittering variance

As mentionned previously, the selection of the jittering variance Σ_ω in Eq. (V.7) is of paramount importance in PF when employing the particle jittering strategy defined in Eq. (V.5). A common choice is to adapt this jittering variance at each step, and not to use a fixed value. [20] proposed an exponentially decreasing jittering variance across time steps, equal for each parameter. The idea is to initially let the particles explore a wide possibility of values in the state space, and then decrease the jittering to focus the exploration on the most suitable trajectories. However, in real case scenarios, and typically with the turbofan example, the beginning of the HI trajectories can be roughly constant around zeros, see Figure V.13, so exploring at the beginning is not a very effective strategy, as the observation can not give much information about the HI trajectory and hence not discriminate between likely and unlikely particles. Moreover, as mentioned before in this section, the proposition in [20] to use a single variance value for each state parameter is clearly sub-optimal and ignores the possible jittering-sensitivity variability among the parameters of the surrogate $\widetilde{g}$. Disregarding the particular use of PF for HI forecasting, some attempts for setting the jittering dynamic have been proposed in the literature. [57] e.g. proposed to update each parameter x_t of $\widetilde{g}$, based on the value's range observed in the current particle set, the number of particles and the number of parameters. [174] refined this strategy with a variance depending also on the parameter value range of the current particles, but also depending on the effective sample size of the PF at the same time step. This last approach in jittering is the one we use in this work. At each time step t, the jittering variance ($\sigma_{\omega,t}^2[d]$ for each component d of any particle $x_t^{(q)}$ is defined as:

$$\sigma_{\omega,t}^2[d] = \widehat{JIQR}_{d;t} ESS_t^{-\frac{1}{3}} \tag{V.13}$$

where $\widehat{IQR}_{d;t}$ is the normalised inter-quartile range of values of d^{th} component of x_t over all the particles, and ESS_t is the effective sample size (ESS) at time t defined as:

$$ESS = \frac{1}{\sum_{q=1}^{N_p} (\hat{w}_t^{(q)})^2} \tag{V.14}$$

where N_p denotes for the number of particles. ESS evaluates the degeneracy of particles. If all particles tend to have similar weights then ESS will tend to N_p, if all particles but a few have weights close to zero, then ESS will tend to this small number of particles with high weights. With this jittering dynamic, a low ESS triggers an increase in jittering variance and thus a greater exploration of the particles, while a high ESS decreases this exploration. In other words, a general decrease in particles likelihood leads to an increasing parameter exploration, while a general increase or steady level of particles likelihood leads to a decreasing or constant parameter exploration. In [174] the authors found a value of $J = 1.59$ based on a general hypothesis. However it is unlikely that this value is optimal for all use cases, therefore J is here set as an hyperparameter for optimising the jittering dynamic. In the present work multiple values have been tried by trial and error and the best performances were obtained with $J = 1.5$.

(e) Additional a priori on particles

The weighting of the particles given their likelihood in Eq. (V.3) ensures the consistency between particles and measurements until the observation time t. To further improve the quality of the prediction, additional a priori can be enforced on the trajectories via filtering the particles. This consists in assigning a zero weight to the particles whose trajectories do not comply to some predefined rules. In this work, we filtered out the particles based on:

Monotonicity
: HIs are by definition monotonous. However, the noise in the observations can depict some non-monotonicity; this can result in a non-monotonous prediction, which is incorrect by essence. This is a monotonicity prior.

Threshold reaching
: Particles are propagated in the future for a fixed horizon, defined as 150% of the longest known HI trajectories. If a particle does not encounter the threshold before the horizon, it is filtered out. This is a prior on the maximum life expectancy.

Max derivative
: The parameterised function family of $\widetilde{g}$, here MLP with exponential activation function, encompasses functions with potential sudden exponential increases that would not fit well with the HI trajectories. To filter these cases out, an a priori is set such that a particle can not have a derivative higher than the highest derivative found in the training set. This is a prior on the maximum degradation rate.

The a priori filtering can be thought as the introduction of prior knowledge in the PF algorithm. It serves to filter out particles that exhibit substantial deviations or are improbable based on this prior distribution.

With this last implementation details of apriori particles filtering, the particle filter based on neural networks named hereafter PF-NN, dedicated to HI forecasting has now been thoroughly detailed. A general flowchart of this method can be found in Figure V.8 for a global overview and understanding.

V–2.1.4 Threshold estimation and RUL prognostic

As mentioned several times, the HI forecasting is not a objective in itself but serves the more general purpose of RUL prognosis and, to reach this purpose, a threshold estimation is necessary. This failure threshold can then be combined with HI prediction to obtain the RUL estimation.

(a) Threshold estimation

As mentioned in Subsection III–2.2 for HI-based RUL prognostic, a HI forecasting procedure must be coupled with a proper setting of the failure threshold. This paragraph is dedicated at outlining the threshold estimation method we propose to couple with the surrogate-PF HI forecasting. The investigation of threshold estimation techniques achieved in Chapter III led to favor the option of an adaptive probabilistic threshold for a better uncertainty quantification of the RUL. The methodology proposed in [26, 117, 208] being the only one found by the authors in that direction, it is thus used and adapted to a PF based HI forecasting. For applying this methodology a set of known complete HI trajectories $\mathcal{HI} = \{HI^{(k)}(t)\}_{k\in\mathcal{K},\, t\in\mathcal{T}_k(T^{(k)})}$ must be available. This hypothesis is verified in the present context. From this set of trajectories, a set of failure thresholds $\mathcal{R} = \{r^{(k)} = HI^{(k)}(T^{(k)})\}_{k\in\mathcal{K}}$ is defined for each trajectory k as the HI value reached at the end of life $HI^{(k)}(T^{(k)})$. At each time step t, the method then relies first on sequentially estimating a weight $\alpha_t^{(k)}$ for each failure threshold $r^{(k)}$ in the set $\mathcal{R}$ corresponding to the similarity between the currently observed trajectory $HI_{0:t}^{(\kappa)}$ and the k^{th} trajectory considered after t steps, $HI_{0:t}^{(k)}$. In their works [26] propose to compute the $\alpha_t^{(k)}$ for each k by solving the following minimisation problem:

$$\alpha_t^{(k)} = \arg \min_{\substack{\{\beta^{(k)}\}_{k\in\mathcal{K}} \\ \beta^{(k)}\in\mathbb{R}_+^*,\, k\in\mathcal{K} \\ \sum_{k\in\mathcal{K}}\beta^{(k)}=1}} \sum_{\tau=0}^{t} (HI_\tau^{(\kappa)} - \sum_{k\in\mathcal{K}} \beta^{(k)} HI_\tau^{(k)})^2 \tag{V.15}$$

Finally, one can combine these weights and their corresponding historical failure thresholds to obtain a distribution of the true failure threshold of the currently observed trajectory $p_{r^{(\kappa)}|HI_{0:t}^{(\kappa)}}$ by e.g. computing the weighted mean and variance of $\mathcal{R}$ and assuming a Gaussian distribution. The main assumption of this approach is that the similarity between two HI trajectories until a certain time t is proportional to the likelihood of them having similar HI values at failure time.

While, in the present work, this existing general method is used, new solutions particularly adapted to PF, are proposed for the weights computation and for the combination of weights and historical failure threshold.

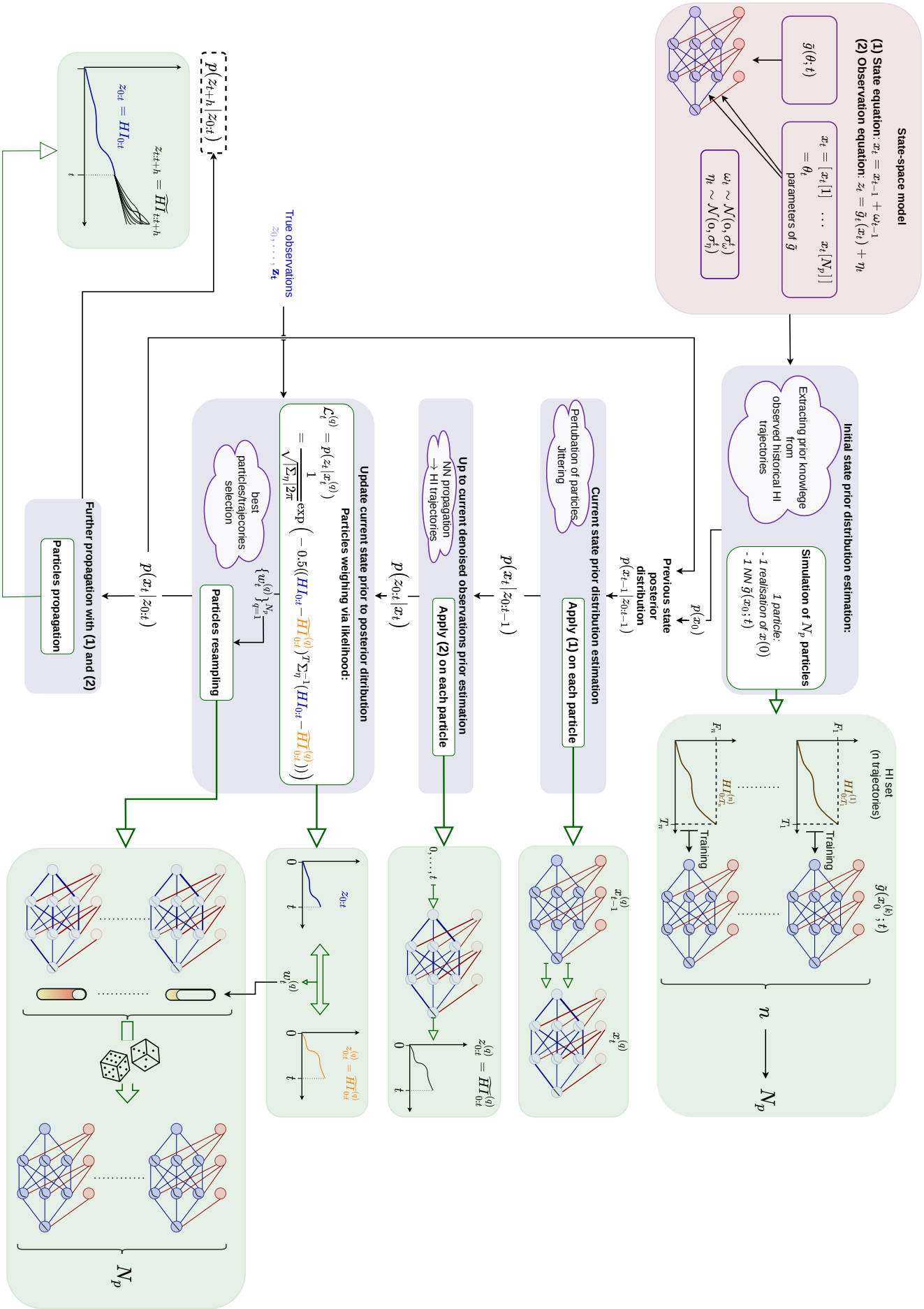


Figure V.8: Flowchart diagram of the proposed PF-NN for HI forecasting

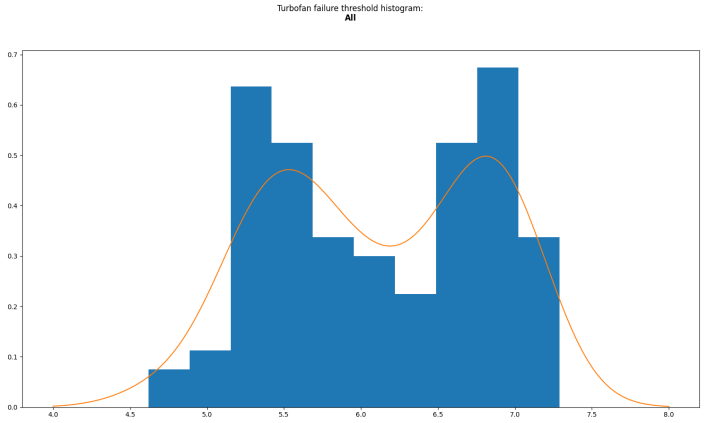


Figure V.9: Histogram of failure thresholds for obtained turbofans HI trajectories and its kernel density estimation at $t = 0$

In a PF, at each step τ, it is trivial to know the respective history of all current particles, that is the state they were in at the previous steps an so on until their first initial particle. The initial particles at $t = 0$ each correspond to one trajectory in $\mathcal{HI}$ fitted by the surrogate function $\widetilde{g}$. We thus denote the number $N_t^{(k)}$ of particles at time t whose initial ancestor correspond to the trajectory k in $\mathcal{HI}$. Because the precise implementation of the PF in this work includes a resampling stage, an initial particle $x^{(q)}$ at step $t = 0$ can have from 0 up to N_p successors at time t. Indeed, if all of any $x_0^{(q)}$ successors are unlikely to fit the observations they are then rarely resampled and end up themselves with almost no successor after a certain number of step. On the contrary if any $x_0^{(q)}$ has successors that are very likely given the observations and that keep being so throughout the time steps, they are then often resampled and this initial particle will have many successors. In this work, we make the hypothesis that $\frac{N_t^{(k)}}{N_p}$ is an estimation of $\alpha_t^{(k)}$ the similarity of the currently observed trajectory and the k^{th} historical HI trajectory up to time t. We thus compute the weights as follows:

$$\alpha_t^{(k)} = \frac{N_t^{(k)}}{N_p} \tag{V.16}$$

Once these weights are acquired they must be combined with their corresponding failure threshold values to obtain a distribution of failure threshold. In this work, no particular hypothesis about the failure threshold distribution family is made (e.g. Gaussian or exponential). This is of paramount importance for the empirical distribution of $\mathcal{R}$ e.g. for the turbofan case Figure V.9. A suitable choice is to estimate the distribution of $\mathcal{R}$ with a weighted kernel density estimation (KDE). This enables to estimate for a current trajectory under observation the density function $p_{r^{(\kappa)}|HI_{0:t}}(r)$ of the current trajectory failure threshold $r^{(\kappa)}$ given the already available observations of this trajectory

$HI_{0:t}^{(\kappa)}$:

$$p_{r^{(\kappa)}|HI_{0:t}^{(\kappa)}}(r^{\kappa}) = \sum_{k=1}^{n} \alpha_t^{(k)} \frac{1}{h} \mathcal{K}(\frac{r^{(\kappa)} - r^{(k)}}{h}),\ r^{(\kappa)} \in \mathbb{R}^+ \tag{V.17}$$

$$\text{where } \sum_{k=1}^{n} \alpha_k(t) = 1 \tag{V.18}$$

$$\text{and } \mathcal{K}(u) = \frac{1}{\sqrt{2\pi}} \exp -\frac{u^2}{2} \tag{V.19}$$

where h is called the bandwidth parameter and is estimated using the Scott's rule [172].

(b) RUL Prediction and confidence intervals

The set of N_p particles at any time t accounts for a representative ensemble of plausible trajectories the currently observed one can take and therefore a representative discrete sample of the probability distribution $p_{\widehat{HI}_{0:\infty}^{(\kappa)}|HI_{0:t}^{(\kappa)})}$. Additionally, the previous paragraph presented how to estimate $p_{r^{(\kappa)}|HI_{0:t}^{(\kappa)}}(r)$ the probability density function of the failure threshold for the currently observed HI trajectory.

For RUL prognosis the quantity of interest can be defined as:

$$L^{(\kappa)} = T^{(\kappa)} - t \tag{V.20}$$

where $L^{(\kappa)}$ and $T^{(\kappa)}$ respectively denotes for the RUL and time of failure of the currently observed trajectory while t is the current time. To estimate the distribution of $L^{(\kappa)}$ we resort to a simulation-based approach. A high number N_s of pairs of trajectories and failure thresholds, $(HI_{0:\infty}^{(j)}, r^{(\kappa)(j)})_{j=1}^{N_s}$ are randomly and independently sampled from the set of current particles $p_{\widehat{HI}_{0:\infty}^{(\kappa)}|HI_{0:t}^{(\kappa)})}$ and the current failure threshold distribution $p(r^{(\kappa)}|HI_{0:t})$. From each of those pairs, a value of $T^{(\kappa)(j)}$ can be obtained as the time needed by $HI_{0:\infty}^{(j)}$ to reach the level $r^{(\kappa)(j)}$. A simple illustration of this process is given in Figure V.10

The set of obtained values $\{T^{(\kappa)(j)}\}_{j=1}^{N_s}$ can then be used to estimate $p(T^{(\kappa)}|HI_{0:t})$. In this work, only the mean and the 90% confidence interval of this latter distribution are of interest and are obtained empirically from $\{T^{(\kappa)(j)}\}_{j=1}^{N_s}$. The RUL $L^{(\kappa)(j)}$ is easily obtained by subtracting the current time t to $T^{(\kappa)(j)}$.

V–2.2 HI learning as pretext task for self-supervised RUL prognosis

V–2.2.1 Self-supervised learning

In Subsection II–2.2 a brief presentation of self-supervised learning (SSL) is given. A more specific version of SSL is also given: contrastive learning. In contrastive learning, knowledge of similarity and dissimilarity between the examples is used to learn a representation of these latter in a latent space of reduced dimension where these known relationships should be emphasised. As a matter of fact, this is precisely the approach that was proposed for learning the HI in Chapter IV. In this method, for encoding samples in a latent space, a prior knowledge of similarity and dissimilarity is used:

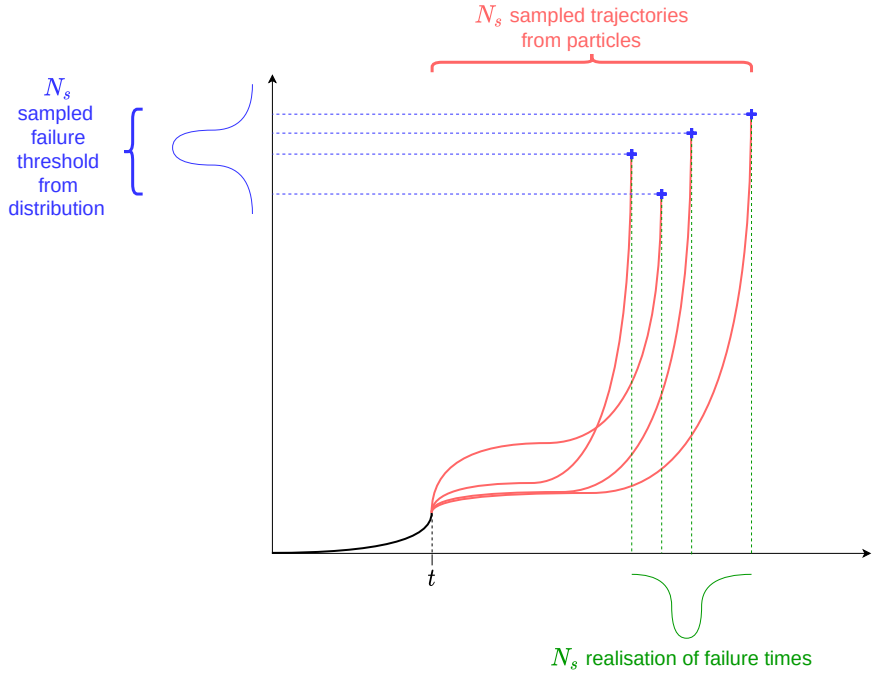


Figure V.10: Illustration of the simulation-based estimation of the failure time distribution

- Two samples should be similar if they come from the same run-to-failure (RtF) and are monitored closely in time.
- The dissimilarity between a sample at the beginning of life and another sample from the same RtF should be greater when this other sample is monitored further away in the future
- The greatest dissimilarity between two samples of the same RtF should be between a sample at the start of life and the sample at the failure time.

In the context of learning a HI in Chapter IV, this latent space is used to build HI trajectories of observed RtFs. For a given RtF, its HI trajectory is the evolution, with the monitoring time of its samples, of their distances in the latent space with the first sample of the same RtF.
In the present context of RUL prognosis, the question that arises is: can the representation of a sample in the latent space learned in Chapter IV be used to directly predict the RUL?

V–2.2.2 Proposed approach: self-supervised RUL prognosis with HI contrastive learning as pretext task

This idea is explored in the present section and is referred to as self-supervised RUL prognosis with HI contrastive learning as pretext task. The aim is then to directly predict the RUL $\widehat{L}_i$ of an instance of the system at a certain time t_i corresponding to the starting time of the sub-series $s^{(k)(i)}$ given its representation in the latent space $e^{(k)(i)} = \text{SNN}(s^{(k)(i)})$.

$$\widehat{L}_i = f(e^{(k)(i)}; \theta_f) \tag{V.21}$$

(V.22)

Where θ_f denotes for the parameters of the learning model f. Of course it is plausible that because of noisy measurements, only the last measured sample is not sufficient for a correct prediction of the RUL. Therefore, a more realistic setting would be to consider a certain amount of samples in the past.

$$\widehat{L}_i = f(e^{(k)(i-h:i)}; \theta_f;) \tag{V.23}$$

where h, the history, denotes for the number of previous samples taken into account for making the prediction.

The choice of the learning model for $f(\theta_f, \cdot)$ now arises. The latent space of which this model will be applied has been learned by a SNN as presented in Chapter IV, and more generally the current work already explored the possibility of integrating neural networks for solving various task of prognostic and health management. A logical choice is therefore to opt for the use of NN once again for modelling $f(\theta_f, \cdot)$.

V–2.2.3 Probabilistic recurrent NN and mean-variance estimation

As explained previously in this chapter, when performing RUL prediction, the evaluation of the prediction uncertainty is of paramount importance for future decision making. To this end, it appears relevant to model $f(\theta_f, \cdot)$ with a probabilistic neural network (PNN). More precisely, we use a mean-variance estimation (MVE) for predicting both a mean prediction of the RUL as well as the variance accounting for the uncertainty of prediction [86]. With this approach the RUL prediction can be written as follow:

$$\widehat{L}_i = \mu_{L_i} + \nu_i \tag{V.24}$$

$$\nu_i \sim \mathcal{N}(0, \sigma_{\widehat{L}_i}) \tag{V.25}$$

where $\mu_{\widehat{L}_i}$ is the mean prediction of the true RUL and ν_i is a zero-centred noise representing the prediction uncertainty.
The output of the PNN f is therefore composed of two values, $\mu_{\widehat{L}_i}$ and $\sigma_{\widehat{L}_i}$ the variance of the noise ν_i:

$$\{\mu_{\widehat{L}_i}, \sigma_{\widehat{L}_i}\}^{(k)} = f(\theta_f, e^{(k)(i-h:i)}) \tag{V.26}$$

To train such a PNN, the loss used is based on the log-likelihood of a Gaussian density probability function with mean and variance predicted by the output of the neural network given the true RUL, denoted hereafter $L_i^{(k)}$.

$$\mathcal{L}(\mu_{\widehat{L}_i}, \sigma_{\widehat{L}_i}, L_i^{(k)}) = -\log(\mathcal{L}(L_i|f; \theta_f; e^{(k)(i-h:i)})) \tag{V.27}$$

$$= -\log(\varphi(\frac{L_i - \mu_{\widehat{L}_i}^{(k)}}{\sigma_{\widehat{L}_i}^{(k)}})) \tag{V.28}$$

$$\tag{V.29}$$

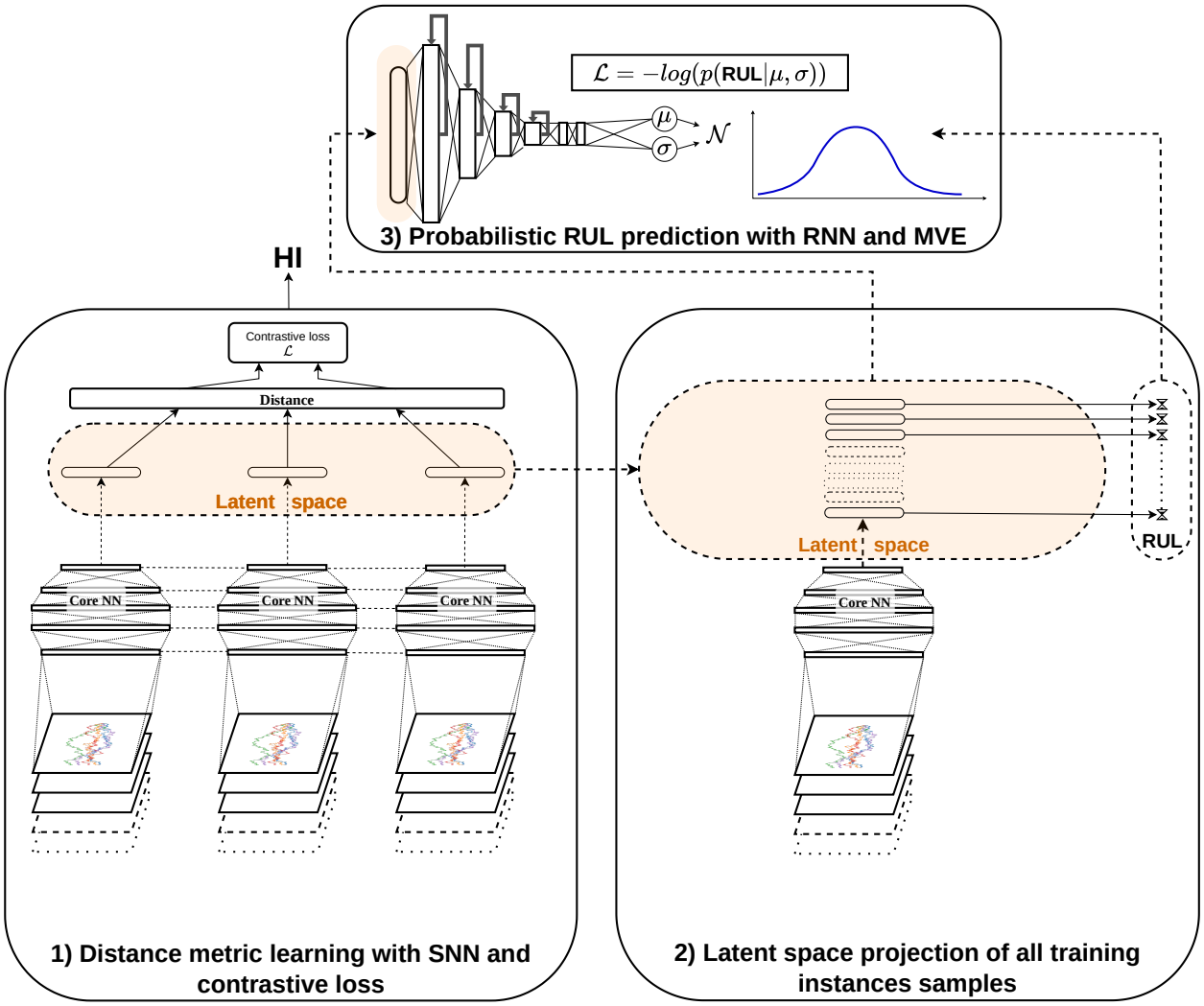


Figure V.11: Overview diagram of the proposed SSL RUL prediction with HI learning as pretext task, named SNN-PRNN

where φ denotes for the PDF of the standard Gaussian distribution, $e^{(k)(i-h:i)}$ is the representation in the latent space of the sub-series $s^{(k)(i)}$, $L_i^{(k)}$ its associated RUL target and where $\mu_{\widehat{L}_i}^{(k)}$ and $\sigma_{\widehat{L}_i}^{(k)}$ are the outputs of the PRNN applied on $e^{(k)(i-h:i)}$. A schematic overview of the method, named hereafter SNN-PRNN, can be found in Figure V.11, and a detailed schematic of the PRNN-MVE architecture in Figure V.12. In this schematic and in the experiment part of this chapter, the history parameter h is set to 10. Such a choice should be more thoroughly investigated for a real use of this approach, to find the right balance between accuracy and computational cost.

To predict the RUL by SNN-PRNN, a simple RNN architecture is here proposed. Because multiple historic steps of samples representation in the latent space are used as input to the model, the chosen architecture is composed of first, recurrent layers, and then, dense layers, with a final dense layer with 2 neurons (one for the mean and one for the variance). It is important noting that because the variance is necessarily positive a softplus activation function is used only on the neuron predicting the variance. For the mean neuron, the sigmoid activation function is used, because the RUL label data used for training the PNN is scaled between $[0, 1]$. One could choose a different architecture with first common layers for mean and variance prediction and additional specific dif-

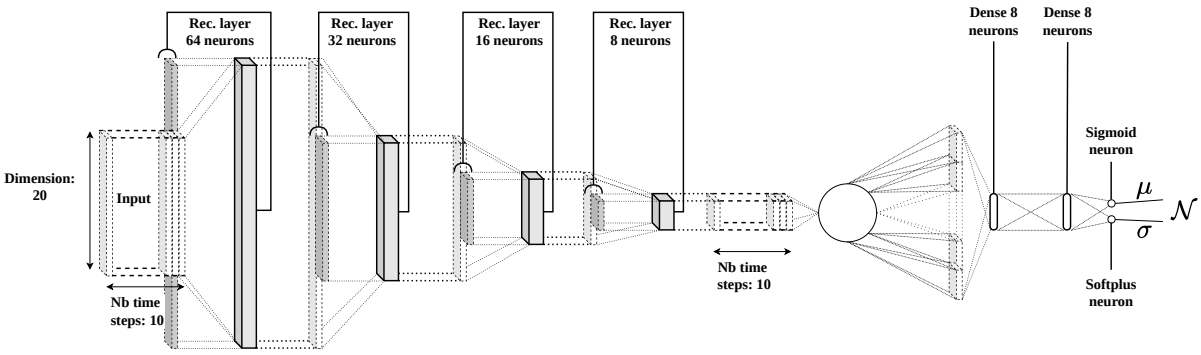


Figure V.12: Proposed PRNN for estimating the mean and standard deviation of the RUL.

ferent layers for each of the two, or one could even decide of two completely different architectures for the mean and variance from the input to the output. The architecture choice in this work is arbitrary and opted for its simplicity, as the aim here is to investigate the possibility of using SNN-PRNN for predicting the RUL of a system instance at any time directly using recent measurement samples representation in the latent space learnt by the methodology presented in Chapter IV.

Once the PRNN is trained, one can easily obtain the RUL mean prediction and 90% CI bounds. The h last samples are fed to the NN, the resulting mean $\mu_{\widehat{L}_i}$ is the RUL prediction and $\mu_{\widehat{L}_i} - 1.96\sigma_{\widehat{L}_i}$ and $\mu_{\widehat{L}_i} + 1.96\mu_{\widehat{L}_i}$ are the CI bounds.

V–2.3 Experiments

The public turbofan engine dataset [167] introduced in Subsection IV–3.1 is used to assess the performance of the proposed methods. Among the 100 engines, the 80 first engines are used as the training set and 20 last ones as the testing set. HI trajectories are here obtained from the siamese neural network proposed in Chapter IV, which is optimised on the training set. Once trained, this HI model is used to produce HI trajectories for each of the engines in the training and testing sets. The obtained HI trajectories of the testing set are drawn in Figure V.13. To evaluate the benefits of the proposed approaches, namely PF-NN and SNN-PRNN, three experiments are conducted. One aims at evaluating the failure threshold estimation in the PF-NN method, another one at evaluating the use of NN as surrogate function instead of more classical surrogate choices in the same NN-PF. Finally a third one evaluating the method of direct SSL RUL with HI learning as pretext task.

(a) Experiment #1

A first experiment was conducted to assess how the PF-NN model performs with different threshold estimation techniques. In the proposed PF-NN method, a new probabilistic threshold estimation technique has been developed. Here, it is proposed to evaluate the PF method using this new threshold estimation method and a classic one, which is setting a constant deterministic threshold as the median of maximum HI values reached by the known complete historical HI trajectories (training set).

(b) Experiment #2

The second experiment that was conducted aimed at assessing the choice of a NN as the surrogate function. Here, it is proposed to evaluate the PF method using the NN as surrogate and a more classic choice found in the literature, which is the sum of two exponentials sometimes denoted double exponential (DE). DE is defined by a sum of two exponential functions as follows:

$$g(t, (a_t, b_t, c_t, d_t)) = a_t \exp b_t t + c_t \exp d_t t \tag{V.30}$$

where $\theta = (a_t, b_t, c_t, d_t)$ are the function parameters and thus the states x_t in the PF state-space equations, see eqs. (V.5) and (V.6). This choice of surrogate model is very common in the literature of HI prediction with PF [101, 145, 79] as mentioned previously. This is due to the degradation dynamics of most of the complex systems that tend to follow an exponential growth. It is in particular verified for the application case studied here, where HI trajectories show an exponential-like behaviour when the degradation appears, see Figure V.13. To initially optimise the DE model on the training HI trajectories a least-square minimisation is performed with the Levenberg-Marquardt algorithm.

(c) Experiment #3

Finally, the third experiment consists in testing the direct RUL prediction methodology of SSL with the HI learning proposed in Chapter IV as pretext task. The results of this method, SNN-PRNN are simply compared to those of the two previous experiments.

In all of these experiments the proposed methods for RUL prediction are performed on each instance of the testing set. The training HI trajectories are used for initialising the particles in the PF-NN method and for training the PRNN model in the SNN-PRNN one.

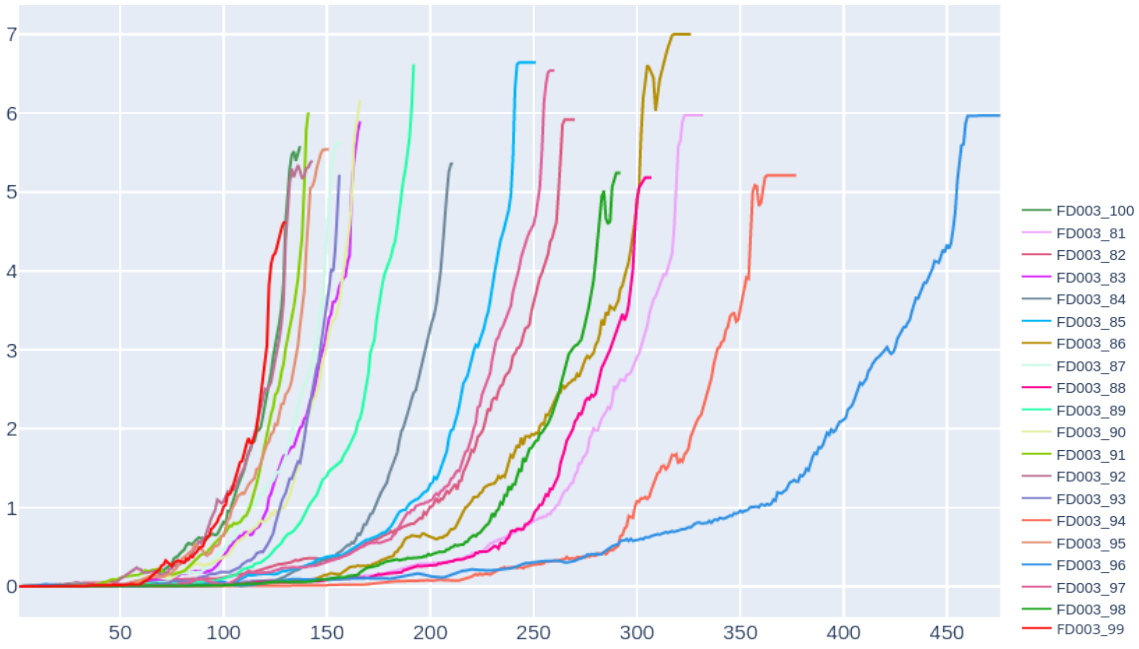


Figure V.13: HI trajectories of the testing set (HI value vs. time)

(d) Evaluation metrics

To assess the performance of the RUL prediction two indicators are computed. The first indicator is the mean relative error (MRE) defined as:

$$MRE = \frac{\sum_{k\in\mathcal{K}}\sum_{t\in\mathcal{T}_k(T^{(k)})}\frac{|\widehat{L_t^{(k)}}-L_t^{(k)}|}{L_t^{(k)}}}{|\mathcal{T}_k(T^{(k)})\mathcal{K}|} \tag{V.31}$$

where k is a particular instance of the system under study, t a particular time step, $\mathcal{T}_k(T^{(k)})$ the set of time steps for which the RUL prediction performance is assessed for the k^{th} instance, $\widehat{L}_t^{(k)}$ the predicted RUL and $L_t^{(k)}$ the true RUL. $|\mathcal{T}_k(T^{(k)})\mathcal{K}|$ here denotes for the total number time steps for all instances that are analysed by the indicator. This indicator relates to the point prediction performance, and needs to be minimised.

A second indicator is developed to assess the performance of the CI prediction. It is referred as CI coverage (CIC) and defined as follow:

$$CIC = \sum_{k\in K}\mathbb{1}_{L_k^*\in\widehat{CI}_k}\frac{L_k^*}{W_{\widehat{CI}_k}} \tag{V.32}$$

where $\mathbb{1}_{L_k^*\in\widehat{CI}_k}$ is the indicator function that takes one if the true RUL lies in the predicted RUL CI, and $W_{\widehat{CI}_k}$ is the CI width. The objective of this indicator, which is to be maximised, is to first reward RUL CIs that include the true RUL, but also to reward more the CIs containing the true RUL that are narrower, with this reward modulated by the true RUL. A wide CI when the true RUL is low is more penalized than when it is high.

To get a better insight on the performances of the tested strategies in all the conducted experiments, their results are analysed over the entire life of the engines and in the last quarter of their life where the RUL prediction is more critical. The results of the online RUL prediction for experiment #1 are shown in Subsection V–2.3 for the five first testing engines. The overall performance indicators for both experiments are summarised in Table V.1, averaged and summed up, respectively for MRE and CIC, over the entire testing set.

(e) Results

Model	MRE (full)	MRE (75-100%)	CI coverage (full)	CI coverage (7
PF-NN + probabilistic threshold	0.516	0.574	3527	1311
PF-NN + constant threshold	0.442	**0.441**	**4299**	1621
PF-DE + constant threshold	**0.429**	0.645	3054	389
SNN-PRNN	0.529	0.953	2040	**1646**

Table V.1: Performance indicators for experiment #1, #2 and #3 over the entire testing set

Experiment #1: It is clear from this experiment that the use of the proposed probabilistic threshold does not lead to improved results, against our expectations: the MRE both on full and last quarter of life indicates better mean prediction performance of the proposed method

with a constant median threshold. However, additional experiments should be performed with more variability in the failure HI values to confirm this finding. For the CI coverage the proposed indicator also indicates better performance with the constant median threshold. These indicator values are crucial for distinguishing the performance differences between the two approaches tested in this experiment. Indeed, the RUL prediction graphs in Subsection V–2.3 show that both approaches predictions does not enable to easily distinguish a clear difference. Nevertheless, these graphs can help us giving insight on the overall RUL prediction quality. They are in one hand encouraging because clearly the RUL prediction systematically converges very close to the true RUL towards the end of life. However these graphs also indicate that the prediction are often overconfident in their predictions leading to confidence intervals excluding the true RUL often for long parts of the engine's life and, in some cases, during the entire life. The uncertainty quantification of the proposed method is therefore not well enough calibrated and necessitates further improvements, as a perspective of this work.

Experiment #2: This second experiment results give a clear indication that using NN as surrogate function improves the performance of the surrogate-PF approach. Indeed, if PF-DE shows slightly better mean RUL prediction accuracy on the entire lifetime of engines, it is clearly outperformed by the PF-NN in the last quarter of life. Moreover, the CIC indicates a consistently better performance of PF-NN for CIC both on full and last quarter of life. PF-DE is even outperformed by the PF-NN with probabilistic threshold, which, as demonstrated by the previous experiment can be a disadvantage compared to the constant median threshold. This result is consistent with the finding of [20] that first proposed to use NN as surrogate function for PF-based HI forecasting in the case of Li-Ion batteries.

Experiment #3: This last experiment indicates that the proposed SNN-PRNN approach leads to poorer result for RUL prediction compared to the PF-based approaches tested. The only satisfaction of this method is its performance of CIC in the last quarter of life which is slightly improved compared to the PF-NN approach with constant threshold. This satisfaction must nevertheless be modulated. Indeed, if the PF-based approaches uncertainty quantification quality is hindered by over-confidence in their predictions, the ones of the SSL approach suffer of under-confidence. This can clearly be observed directly on the RUL prediction graph resulting from the third experiment in Subsection V–2.3. The true RUL is systematically included in the 90% CI interval, which might indicate that the RUL prediction distribution, under the gaussian assumption here, is miscalibrated, leading to a too wide CI interval. This CI prediction is nevertheless better rewarded by CIC than the PF-based ones. This is because these latter, more often than expected, exclude the true RUL from the CI prediction towards the end of life, an outcome purportedly penalised by the proposed CIC, and never observed with the SNN-PRNN approach. One should keep in mind that this method has not been optimised thoroughly compared to the PF-NN one, but is more an exploratory work.

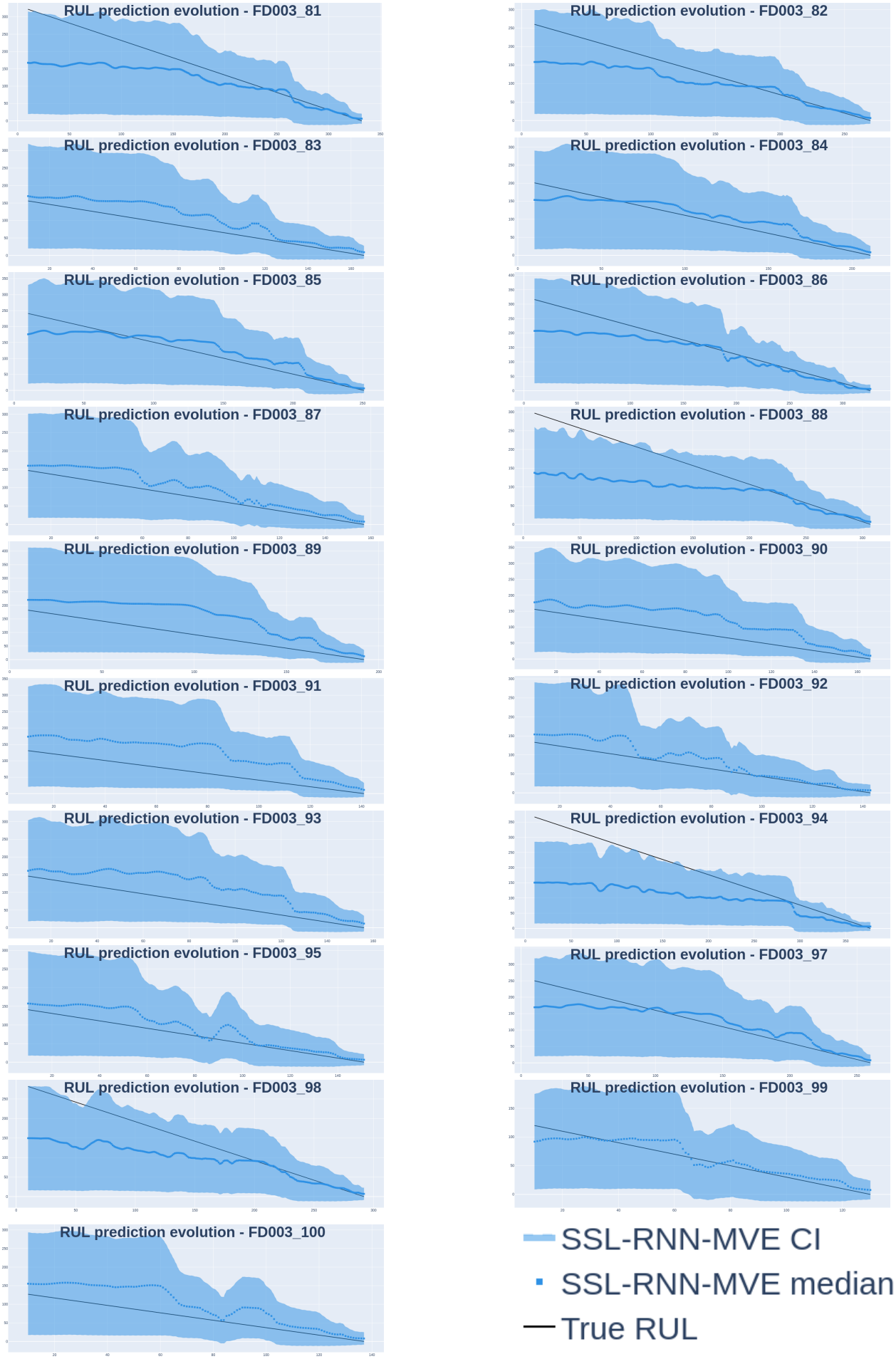


Figure V.14: RUL prediction graph of SNN-PRNN on testing turbofan engines (RUL vs. time)

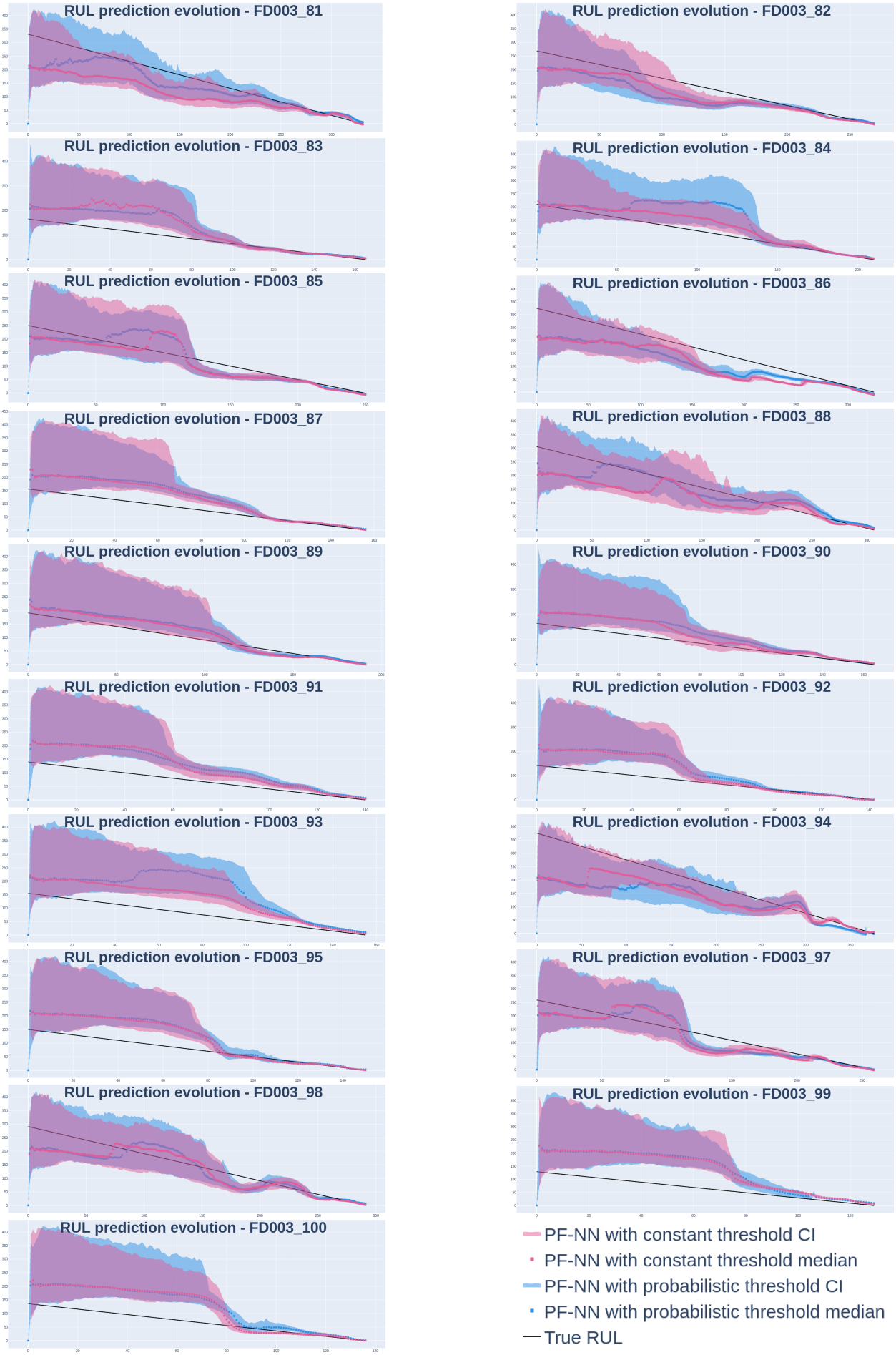


Figure V.15: RUL prediction graph, comparison of PF-NN with probabilistic and constant threshold on testing turbofan engines (RUL vs. time)

V–3 Conclusion and perspectives

V–3.1 Conclusions

This final chapter aimed at exploring the possibility of using the information extracted in Chapter IV, that is: the HI trajectories, and the latent space produced by the SNN with which the HI trajectories are learnt. These information are investigated to be used for both RUL prognostic and failure mode identification.

The two proposed methods explored for RUL prognostic lead to interesting conclusion.
In the first one, PF-NN, a combination of particle filter and ensemble of neural network is used for predicting the RUL. This approach extends the work of [20], however, if the general idea does come from the latter reference, new additional considerations have been developed in the present chapter. They include the following: the particle initialisation, the particle exploration dynamic (jittering), the a-priori particle filtering and the combination of the approach with a probabilistic evolving failure threshold. These improvements led to extend the finding of [20], on a new application case, that simple shallow neural network are particularly adapted surrogate function for particle filter with unknown state and observation equations. Indeed, the PF-NN strategy was applied to simulated turbofans were the degradation is exponential like, an we compared results with neural network instead of sum of exponential models. The neural networks led to better RUL prediction. However, we also found that in this application case the uncertainty quantification is miscalibrated and leads to overconfident prediction, despite being better calibrated than the method of sum of exponential.

As a second RUL prognostic approach we propose the SNN-PRNN method: the use of the SNN's learnt latent space for direct and supervised RUL prediction, thus placing ourselves in a self-supervised learning setting. If the RUL prediction results are less satisfactory than the one issued from the first proposed approach, the comparison between the two approaches must be tempered. In the first approach at any current time t the RUL prediction model can use all the observations from initial time up to t. In the second approach the model use only the 10 last observations which is an arbitrary choice. Inversely, in the first approach the model has only access to the HI values (single dimension) while in the second approach the model has access to the entire sample projection in the latent space (multiple dimensions). Interestingly, this second approach seems to lead to better uncertainty quantification quality toward the end of life which is definitely an appreciated outcome for RUL prognostic. However, as opposed to the overconfidence of the particle filter approach prediction, the prediction issued from self-supervised approach is largely under-confident.

Finally in the first section an exploratory study of using HI trajectories for failure mode identification was performed. It was found that a simple univariate time series clustering was able to partially separate instances affected by different failure modes into different clusters. If this separation is not perfect, it is still encouraging. This proposition is merely exploratory in the present thesis and is not a claim that failure mode identification should strictly be done based on HI trajectories. It is rather a suggestion to exploit information contained in HI trajectories to improve a potential failure mode identification task.

VI Conclusion

This chapter finally concludes this thesis by summarising the key developments carried out and the associated findings. It also outlines the current limitations of the proposed developments and the future perspectives of research they offer.

VI–1 Summary of the thesis

This thesis aims at studying multivariate time series (MTS) learning methods in the context of prognostics and health management (PHM). A general presentation of PHM has thus been outlined in Chapter I. The four main tasks of PHM have been presented: anomaly detection, failure mode identification, remaining useful life (RUL) prognostic and health assessment. It has been argued that this latter is connected to all the other tasks. Indeed, by extracting information about the precise health status of a system at any monitored instant in the form of an health indicator (HI), one should be able to better achieve the goals of the three other tasks: RUL prognostic, anomaly detection and failure mode identification. Based on this analysis, the core of this PhD thesis was then more explicitly defined as exploring the analysis of PHM data focusing on health assessment. More precisely the research has been structured in two main axes. First, researching the current limitations of this health assessment task and propose solutions to overcome them. Secondly, researching the current limitations in the usage of HI for subsequent usage in solving other PHM tasks, and also propose solutions to overcome them.

Before the exploration of the literature carried out in Chapter III on the two identified axes of research, Chapter II introduces key theoretical concepts, for a correct understanding of the developments carried out in subsequent chapters. This includes precise definitions of time series, their mathematical notations and particular properties with a focus on different types of sampling used in the following chapters. Then, an introduction to machine learning methods in general and for solving particular problems encountered when learning from MTS has been given. Finally a short focus on theoretical elements of neural networks has been carried out.

In Chapter III, an overview of the literature on HI definition, learning and usage in the field of PHM has been outlined. It includes the definition of what a HI is and the mathematical definitions of what properties this latter must satisfy. The properties of monotonicity, failure consistency, prognosability and robustness have then been emphasised as the most crucial properties for a HI to satisfy, but it was also demonstrated that the property of monotonicity stands out of the three other

ones as even more crucial. Then, existing approaches for learning a HI from preprocessed signals have been described. During this analysis of the state-of-the-art on HI learning approaches, the ability of each approach to enforce monotonicity has been scrutinized . It turns out that, a certain type of approach, namely **similarity-based** approach, has almost never been subjected to developments in the direction of enforcing monotonicity during the learning of the HI. This is in constrast with the other main type of approach identified: **fusion-based** approach that already have been successfully developed to enforce monotonicity. These observations lead to the development of a proposed solution for similarity-based HI learning with a monotonicity constraint in Chapter IV.
In the second part of this chapter, a review of existing works that make use of HI for solving other PHM tasks has been given. It turns out that it is mostly the RUL prognostic task that resorts to HI. The task of anomaly detection can also benefit from HI, but works in that direction have not been included in this chapter for the reason that anomaly detection was not focused on during this thesis. The reason for this choice, as previously explained in Chapter I, is that once a HI is obtained, anomaly detection merely consist in defining a threshold that, when crossed by the HI, indicates an anomaly. This threshold estimation task is similar to the one developed in RUL prognostic. Hence, This chapter's part has been more focused on outlining state-of-the-art approaches for HI-based RUL prognostics. This includes the forecasting of future values of a currently observed HI, and the estimation of a failure threshold which, when crossed by the HI, indicates failure.
It has been pointed out that some promising RUL prognostic may be obtained when mixing stochastic process modeling with ML approaches. Indeed, it is shown is this chapter that stochastic process modeling provides with strong statistical methods for estimating uncertainties in prediction but lack the abilities to precisely model the potential evolutions of HI trajectories issued from systems with complex degradation dynamics. The neural networks on the contrary have been presented in this chapter to have strong ability to precisely model these degradation. These observations lead to propose improvements to a type of approach, in Chapter V, that recently emerged in the literature. This latter combines a stochastic process modeling technique, state-space models and neural networks. Finally, an other interesting finding in this chapter was the absence of any HI-based approach for failure mode identification which also lead us to explore the possibility of using HI trajectories to identify failure modes in Chapter V.

In Chapter IV, to overcome the lack of HI learning method that enforces monotonicity discovered in the literature in the first part of Chapter III, a new approach is proposed. It is built upon a contrastive learning setting with a siamese neural network (SNN) as the representation learning model. Two distinct training sample selections and constraints incorporated directly in the contrastive loss function are proposed to enforce the monotonicity of the resulting similarity-based HI. This approach has the advantage of not resorting to particular hypothesis on the evolution of the HI, which is a significant novelty from the literature and makes the proposed method very generic and theoretically adaptable to any physical system. This method has been tested on two public datasets. The first one, on turbofan, shows the efficiency of this method for enforcing the monotonicity on a dataset with relatively high dimension in the MTS and also a high number of system's instances. A clear correlation between the improvement of the loss of the model and the resulting monotonicity of the HI has been established for the two distinct strategies of monotonicity constraints proposed in this chapter. The second dataset, on rolling bearings, shows the efficiency of the proposed model on reaching its aim for datasets with a reduced number of system's instances, where the MTS are sampled at high-frequency, and whose system degradation dynamics are known to be complex and

challenging to learn. The proposed method has been compared to another similarity-based HI learning method that also tries to enforce monotonicity. It turned out that our proposed methodology performed better on the bearing dataset. The results from the experiments on these two datasets, tend to prove the high adaptability of the proposed methodology for different contexts of data, low or high number of dimensions, low or high frequency sampling, low or high number of system's instances, and different MTS preprocessing techniques. But it also opens perspectives as described later in the dedicated section of this conclusion.

Chapter V aims at exploring the possibility of using the outcomes of Chapter IV, the HI trajectories and the latent space produced by the SNN with which the HI trajectories are learnt. These outcomes have been investigated to be used for both RUL prognostic and failure mode identification. First, in this chapter, an exploratory study has been conducted on the use of HI trajectories for failure mode identification. It has been found that simple univariate time series clustering is able to partially separate instances affected by different failure modes into different clusters. Although this separation is not perfect, it is an encouraging work in that direction. This suggests that it is possible to use the information contained in the HI trajectories to support a potential failure mode identification task. Subsequently, two proposed methods have been explored for HI-based RUL prognostic.

In the first one, PF-NN, a combination of particle filter (PF) and ensemble of neural network has been investigated for RUL prognostic. In this approach a state-space model partially defined by a neural network is continuously optimised by a PF algortithm. This latter continuously update the parameters of pre-trained NNs to fit the current partial HI trajectory being observed in order to forecast it. This method also includes failure threshold estimation to define the HI value that indicates failure of the system. This approach comes from previous works in the literature, however, in this chapter, new additional considerations have been developed for it. These latter include the following: the particle initialisation, the particle exploration dynamic (jittering), the a-priori particle filtering and the combination of the approach with a probabilistic evolving failure threshold. These improvements lead to extend previous findings in the literature that simple shallow neural network are particularly adapted surrogate function for particle filter with unknown state and observation equations. The PF-NN strategy has here been applied to simulated turbofans were the degradation is exponential like, and has been compared with a similar PF approach that does not use NN. The neural networks lead to better RUL prediction. However, it has also been found that, in this application case, the model provides overconfident prediction, which is a sign of low quality of uncertainty quantification.

A second RUL prognostic approach has also been proposed, the SNN-PRNN method. Here the SNN's learnt latent space has been used for direct and supervised RUL prediction via a probabilistic recurrent neural network (PRNN), thus placing this approach in a self-supervised learning setting. The RUL prediction results are less satisfactory than the ones issued from the PF-NN approach but the comparison between the two approaches must be tempered. The two approaches use different inputs and take into considerations different amount of previous observations for predicting the RUL. Additionally, the second approach is a completely novel proposition and therefore an exploratory work here, when the first proposed approach has been built upon previous existing works and benefits from the accumulation of improvements from many researchers.

VI–2 Limitations and perspectives

This thesis developments and findings will advance the field of PHM. The new similarity-based HI learning method using SNN and contrastive loss does compensate for a lack of similarity-based HI method that enforces monotonicity of the HI in the literature. This method provides one with a HI more compliant with the theoretical properties it should satisfy, and with a latent space where samples of monitored signals can be somehow represented according to their degradation levels. The PF-NN methodology enhanced in this thesis will enable researchers to better combine stochastic process modelling and neural networks for forecasting HI values of a current trajectory and predicting the RUL. Finally, this thesis proposes interesting perspectives in using HI trajectories in general for failure mode identification, and using HI learning as a pretext task before applying a supervised direct RUL prediction in a self-supervised learning setting.

However, a thesis being limited in time, its developments also have limitations. Hence, the proposed methods in Chapter IV and Chapter V are subjected to improvements for future works.

VI–2.1 Perspectives on HI learning with SNN and contrastive loss

Firstly, the impact of the number of failure modes and run-to-failure samples could be analysed more in depth. Here two datasets have been studied, the one of turbofans with a high number of instances and low number of failure modes, and the one of bearing with an unknown, but probably high, number of failure modes, and low number of instances. The variety of failure modes may induce a high variability in the data and thus necessitates a higher number of system's instances for a model to be able to generalise well. A thorough analysis of the impact of lowering or increasing the number of available instances in different data availability contexts would be an interesting addition to the present work, as well as any proposed approach for HI construction models. More existing datasets could therefore be studied with the proposed approach, e.g. Li-Ion battery dataset from the same NASA repository, in order to confirm the obtained results and study the sensitivity to the number of instances and number of failure modes.

An interesting usage of the learnt distance metric, would also be to study the distances between samples of different instances of the system. Indeed, in the proposed approach for HI construction, once the distance metric is learnt, only samples of the same instance are compared. But instead of comparing any sample to the initial sample of the same instance, one could compare it to all initial samples of every known instances for a better HI estimation. One could even compute an entire distance matrix between any observed samples of any observed instance. This distance matrix could be used to project every samples relatively to each other in a reduced dimension (2D or 3D) thanks to manifold projection techniques like t-SNE, LLE or Isomap. These manifold techniques could be very useful for visualising the different degradation trajectories or even for discovering regions of the manifolds indicating different stages of degradation or different failure modes.

Another potential for improvement relates to uncertainty quantification (UQ). This topic is often absent of the problem of HI construction and has not been prioritised in this chapter. Indeed, usually the issue of UQ is rather accounted for during the RUL prediction. An interesting perspective would then be to construct HI not in a deterministic form, i.e. a single trajectory, but in a probabilistic form, i.e. as a distribution of trajectories, or a stochastic process. In the continuity of this work, investigating the possibility of transforming the current SNN based model into a probabilistic one would be a promising perspective. An interesting work in that direction is the one of [205] that develops a probabilistic combination of triplet loss and SNN, with the use of Bayesian NN and mean-variance estimation for uncertainty quantification. This work has been developed in the context of image retrieval, but some key concepts could be reused in the present context.

VI–2.2 Perspectives in using HI learning for RUL prediction and failure mode identification

The results on the experiment of failure mode identification suggest to further investigate the use of HI trajectories for this task. Using the proposed failure mode identification via the HI trajectories could also potentially improve the further RUL prediction, by taking into account the predicted failure mode. Indeed, knowing an observed instance to be subjected to particular failure mode will enable the RUL prediction model to use known HI trajectories from the same failure modes and hence potentially provides with more adapted and precise RUL predictions. This could also improve the failure threshold estimation accuracy, as the HI value at failure time might be dependent on the failure mode.

The PF-NN RUL prognostic approach proposed in Chapter V could benefit from a thorough ablation study to investigate the respective impacts of the improved jittering dynamic, particles initialisation and a-priori particle filtering on the quality of the results. A valuable addition would also be to experiment on different RUL datasets as for instance the one on bearings used in Chapter IV. Due to the time constraint of the thesis the experiments of PF-NN on the bearing dataset have not been possible to carry out satisfactorily and are hence not included here. Another dataset that could be used is the one on Li-ion batteries as it is the one used by the authors of the original formulation of PF-NN that inspired the proposed method. It could enable to evaluate our improvements against this original proposed approach.

For the SNN-PRNN approach for RUL prediction proposed in Chapter V, many suggestions may be raised. For now, the approach attempts at predicting the RUL of any instance a certain time via a PRNN, inputted with the last 10 observations, up to the current time, of the SNN's latent space projection of measured signals samples. Thus, a simple improvement could be to include as input to the PRNN the time elapsed from the beginning of life until the last time of observations, which can be a crucial information to more accurately predict the RUL. Studying the impact of reducing or increasing the number of last observations to take into account for predicting the RUL would also be a necessary work to obtain a better model for RUL prediction. Finally, studying the

use of more complex NN architecture, particularly adapted to treat sequential data, like dilated convolution layers or attention mechanisms could also improve the approach.

Finally, the results from the two proposed RUL prognostic approach also highlight that quantifying the prediction uncertainties remains a challenge and should be prioritised as an axis for future research. For this, it would be relevant to develop models that have the ability to distinguish between the aleatoric and epistemic level of uncertainties produced by the RUL prediction. These two types of uncertainties are not explicitly accounted for in this thesis, however, this aspect offers interesting perspectives especially for the second proposed RUL prognostic approach SNN-PRNN. Indeed the mean-variance estimation developed in this approach can indeed be thought of as accounting for aleatoric uncertainty, but integrating uncertainties on the NN parameters could enable to include epistemic uncertainty by e.g. resorting to bayesian neural networks.

VI–3 Epilogue

This manuscript attempted to summarise the work carried out during the my thesis. It aimed at exploring the use of multivariate time series learning and analysis in the field of prognostic and health management with a particular focus on the use of neural networks and deep learning tools. I hope this work will find echos in researchers of this field, and inspire them to reuse, improve and study the proposed methods of HI learning, RUL prognostic and failure mode identification. I would like to note that the developed method for solving PHM tasks made use of generic machine learning approaches and contribute with some advancements in this field as well. The proposed approaches could hence also be used in other contexts than PHM: contexts where enforcing a monotonicity constraint in a contrastive learning setting on sequential data is necessary, or contexts where on-line forecasting of a monotonous quantity of interest is needed.

Bibliography

[1] S. Abbasion et al. "Rolling Element Bearings Multi-Fault Classification Based on the Wavelet Denoising and Support Vector Machine." In: *Mechanical Systems and Signal Processing* 21.7 (Oct. 2007), pp. 2933–2945. ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2007.02.003.

[2] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. "Time-Series Clustering – A Decade Review." In: *Information Systems* 53 (Oct. 2015), pp. 16–38. ISSN: 03064379. DOI: 10.1016/j.is.2015.04.007.

[3] A. S. Ahmad et al. "A Review on Applications of ANN and SVM for Building Electrical Energy Consumption Forecasting." In: *Renewable and Sustainable Energy Reviews* 33 (May 2014), pp. 102–109. ISSN: 1364-0321. DOI: 10.1016/j.rser.2014.01.069.

[4] Wasim Ahmad et al. "A Reliable Technique for Remaining Useful Life Estimation of Rolling Element Bearings Using Dynamic Regression Models." In: *Reliability Engineering & System Safety*. Impact of Prognostics and Health Management in Systems Reliability and Maintenance Planning 184 (Apr. 2019), pp. 67–76. ISSN: 0951-8320. DOI: 10.1016/j.ress.2018.02.003.

[5] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. "A Survey of Network Anomaly Detection Techniques." In: *Journal of Network and Computer Applications* 60 (Jan. 2016), pp. 19–31. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2015.11.016.

[6] Masanao Aoki. *State Space Modeling of Time Series*. Springer Science & Business Media, Mar. 2013. ISBN: 978-3-642-75883-6.

[7] M.S. Arulampalam et al. "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking." In: *IEEE Transactions on Signal Processing* 50.2 (Feb. 2002), pp. 174–188. ISSN: 1941-0476. DOI: 10.1109/78.978374.

[8] Vepa Atamuradov et al. "Railway Point Machine Prognostics Based on Feature Fusion and Health State Assessment." In: *IEEE Transactions on Instrumentation and Measurement* 68.8 (Aug. 2019), pp. 2691–2704. ISSN: 0018-9456, 1557-9662. DOI: 10.1109/TIM.2018.2869193.

[9] P. Baraldi, G. Bonfanti, and E. Zio. "Differential Evolution-Based Multi-Objective Optimization for the Definition of a Health Indicator for Fault Diagnostics and Prognostics." In: *Mechanical Systems and Signal Processing* 102 (Mar. 2018), pp. 382–400. ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2017.09.013.

[10] Ole E. Barndorff-Nielsen, Thomas Mikosch, and Sidney I. Resnick. *Lévy Processes: Theory and Applications*. Springer Science & Business Media, Mar. 2001. ISBN: 978-0-8176-4167-2.

[11] Luis Basora, Xavier Olive, and Thomas Dubot. "Recent Advances in Anomaly Detection Methods Applied to Aviation." In: *Aerospace* 6.11 (Nov. 2019), p. 117. ISSN: 2226-4310. DOI: 10.3390/aerospace6110117.

[12] Satyanarayana Reddy Beeram and Swarna Kuchibhotla. "Time Series Analysis on Univariate and Multivariate Variables: A Comprehensive Survey." In: *Communication Software and Networks*. Ed. by Suresh Chandra Satapathy et al. Lecture Notes in Networks and Systems. Singapore: Springer, 2021, pp. 119–126. ISBN: 9789811553974. DOI: 10.1007/978-981-15-5397-4_13.

[13] T. Benkedjouh et al. "Remaining Useful Life Estimation Based on Nonlinear Feature Reduction and Support Vector Regression." In: *Engineering Applications of Artificial Intelligence* 26.7 (Aug. 2013), pp. 1751–1760. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2013.02.006.

[14] Maximilian Benker, Artem Bliznyuk, and Michael F. Zaeh. "A Gaussian Process Based Method for Data- Efficient Remaining Useful Life Estimation." In: *IEEE Access* 9 (2021), pp. 137470–137482. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3116813.

[15] Gérard Biau and Erwan Scornet. "A Random Forest Guided Tour." In: *TEST* 25.2 (June 2016), pp. 197–227. ISSN: 1863-8260. DOI: 10.1007/s11749-016-0481-7.

[16] *Box and Jenkins: Time Series Analysis, Forecasting and Control | SpringerLink*. https://link.springer.com/chapter/10.1057/9781137291264_6.

[17] Janet M. Box-Steffensmeier et al. *Time Series Analysis for the Social Sciences*. Analytical Methods for Social Research. Cambridge: Cambridge University Press, 2014. ISBN: 978-0-521-87116-7. DOI: 10.1017/CBO9781139025287.

[18] Rasmus Bro and Age K. Smilde. "Principal Component Analysis." In: *Analytical Methods* 6.9 (2014), pp. 2812–2831. DOI: 10.1039/C3AY41907J.

[19] Jane Bromley et al. "Signature Verification Using a "Siamese" Time Delay Neural Network." In: *Advances in Neural Information Processing Systems*. Vol. 6. Morgan-Kaufmann, 1993, p. 8.

[20] F. Cadini et al. "State-of-Life Prognosis and Diagnosis of Lithium-Ion Batteries by Data-Driven Particle Filters." In: *Applied Energy* 235 (Feb. 2019), pp. 661–672. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2018.10.095.

[21] Francesco Cadini et al. "Particle Filtering-Based Adaptive Training of Neural Networks for Real-Time Structural Damage Diagnosis and Prognosis." In: *Structural Control and Health Monitoring* 26.12 (2019), e2451. ISSN: 1545-2263. DOI: 10.1002/stc.2451.

[22] Fatih Cakir et al. "Deep Metric Learning to Rank." In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, June 2019, pp. 1861–1870. ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.00196.

[23] Yudong Cao et al. "A Novel Temporal Convolutional Network with Residual Self-Attention Mechanism for Remaining Useful Life Prediction of Rolling Bearings." In: *Reliability Engineering & System Safety* 215 (Nov. 2021), p. 107813. ISSN: 0951-8320. DOI: 10.1016/j.ress.2021.107813.

[24] Stanislas Chambon et al. "A Deep Learning Architecture for Temporal Sleep Stage Classification Using Multivariate and Multimodal Time Series." In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 26.4 (Apr. 2018), pp. 758–769. ISSN: 1558-0210. DOI: 10.1109/TNSRE.2018.2813138.

[25] S.G. Chang, Bin Yu, and M. Vetterli. "Adaptive Wavelet Thresholding for Image Denoising and Compression." In: *IEEE Transactions on Image Processing* 9.9 (Sept./2000), pp. 1532–1546. ISSN: 10577149. DOI: 10.1109/83.862633.

[26] Abdallah Chehade, Scott Bonk, and Kaibo Liu. "Sensory-Based Failure Threshold Estimation for Remaining Useful Life Prediction." In: *IEEE Transactions on Reliability* 66.3 (Sept. 2017), pp. 939–949. ISSN: 0018-9529, 1558-1721. DOI: 10.1109/TR.2017.2695119.

[27] Dingliang Chen et al. "Health Indicator Construction by Quadratic Function-Based Deep Convolutional Auto-Encoder and Its Application into Bearing RUL Prediction." In: *ISA Transactions* 114 (Aug. 2021), pp. 44–56. ISSN: 0019-0578. DOI: 10.1016/j.isatra.2020.12.052.

[28] Jiayu Chen et al. "A Novel Health Indicator for PEMFC State of Health Estimation and Remaining Useful Life Prediction." In: *International Journal of Hydrogen Energy* 42.31 (Aug. 2017), pp. 20230–20238. ISSN: 0360-3199. DOI: 10.1016/j.ijhydene.2017.05.241.

[29] Longting Chen et al. "Health Indicator Construction of Machinery Based on End-to-End Trainable Convolution Recurrent Neural Networks." In: *Journal of Manufacturing Systems* 54 (Jan. 2020), pp. 1–11. ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2019.11.008.

[30] Tianqi Chen and Tong He. "Xgboost: eXtreme Gradient Boosting." In: ().

[31] Davide Chicco. "Siamese Neural Networks: An Overview." In: *Artificial Neural Networks*. Ed. by Hugh Cartwright. Methods in Molecular Biology. New York, NY: Springer US, 2021, pp. 73–94. ISBN: 978-1-07-160826-5. DOI: 10.1007/978-1-0716-0826-5_3.

[32] Edwin K P Chong. "An Introduction to Optimization." In: ().

[33] William S. Cleveland. "Robust Locally Weighted Regression and Smoothing Scatterplots." In: *Journal of the American Statistical Association* 74.368 (Dec. 1979), pp. 829–836. ISSN: 0162-1459. DOI: 10.1080/01621459.1979.10481038.

[34] Jamie Coble and J. Wesley Hines. "Identifying Optimal Prognostic Parameters from Data: A Genetic Algorithms Approach." In: *Annual Conference of the PHM Society* 1.1 (2009). ISSN: 2325-0178.

[35] *Computational Intelligence in Time Series Forecasting*. Advances in Industrial Control. London: Springer-Verlag, 2005. ISBN: 978-1-85233-948-7. DOI: 10.1007/1-84628-184-9.

[36] Noel Cressie. "The Origins of Kriging." In: *Mathematical Geology* 22.3 (Apr. 1990), pp. 239–252. ISSN: 1573-8868. DOI: 10.1007/BF00889887.

[37] Padraig Cunningham, Matthieu Cord, and Sarah Delany. "Supervised Learning." In: Jan. 2008, pp. 21–49. ISBN: 978-3-540-75170-0. DOI: 10.1007/978-3-540-75171-7_2.

[38] Marco Cuturi and Arnaud Doucet. *Autoregressive Kernels For Time Series*. Jan. 2011. DOI: 10.48550/arXiv.1101.0673. arXiv: 1101.0673 [stat].

[39] Perry de Valpine. "Review of Methods for Fitting Time-Series Models with Process and Observation Error and Likelihood Calculations for Nonlinear, Non-gaussian State-Space Models." In: *Bulletin of Marine Science* 70.2 (Mar. 2002), pp. 455–471.

[40] K. C. Deekshit Kompella, Mannam Venu Gopala Rao, and Rayapudi Srinivasa Rao. "Bearing Fault Detection in a 3 Phase Induction Motor Using Stator Current Frequency Spectral Subtraction with Various Wavelet Decomposition Techniques." In: *Ain Shams Engineering Journal* 9.4 (Dec. 2018), pp. 2427–2439. ISSN: 2090-4479. DOI: 10.1016/j.asej.2017.06.002.

[41] Li Deng, Geoffrey Hinton, and Brian Kingsbury. "New Types of Deep Neural Network Learning for Speech Recognition and Related Applications: An Overview." In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. May 2013, pp. 8599–8603. DOI: 10.1109/ICASSP.2013.6639344.

[42] Sachin Desale et al. "Heuristic and Meta-Heuristic Algorithms and Their Relevance to the Real World: A Survey." In: *INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING IN RESEARCH TRENDS* 2.5 (2015).

[43] Reik V. Donner et al., eds. *Nonlinear Time Series Analysis in the Geosciences: Applications in Climatology, Geodynamics and Solar-Terrestrial Physics*. Vol. 112. Lecture Notes in Earth Sciences. Berlin, Heidelberg: Springer, 2008. ISBN: 978-3-540-78937-6 978-3-540-78938-3. DOI: 10.1007/978-3-540-78938-3.

[44] Marco Dorigo and Thomas Stützle. "Ant Colony Optimization: Overview and Recent Advances." In: *Handbook of Metaheuristics*. Ed. by Michel Gendreau and Jean-Yves Potvin. International Series in Operations Research & Management Science. Cham: Springer International Publishing, 2019, pp. 311–351. ISBN: 978-3-319-91086-4. DOI: 10.1007/978-3-319-91086-4_10.

[45] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. "On Sequential Monte Carlo Sampling Methods for Bayesian Filtering." In: *Statistics and Computing* 10.3 (July 2000), pp. 197–208. ISSN: 1573-1375. DOI: 10.1023/A:1008935410038.

[46] Kathryn Anne Dowsland and Jonathan Thompson. "Simulated Annealing." In: *Handbook of Natural Computing*. Ed. by Grzegorz Rozenberg, Thomas Back, and Joost N. Kok. Springer-Verlag, July 2012, pp. 1623–1655. ISBN: 978-3-540-92909-3. DOI: 10.1007/978-3-540-92910-9_49.

[47] Elhoussin Elbouchikhi et al. "An Efficient Hilbert–Huang Transform-Based Bearing Faults Detection in Induction Machines." In: *IEEE Transactions on Energy Conversion* 32.2 (June 2017), pp. 401–413. ISSN: 1558-0059. DOI: 10.1109/TEC.2017.2661541.

[48] Sascha M. M. Fässler et al. "Does Larval Mortality Influence Population Dynamics? An Analysis of North Sea Herring (Clupea Harengus) Time Series." In: *Fisheries Oceanography* 20.6 (2011), pp. 530–543. ISSN: 1365-2419. DOI: 10.1111/j.1365-2419.2011.00600.x.

[49] Olga Fink, Enrico Zio, and Ulrich Weidmann. "Fuzzy Classification With Restricted Boltzman Machines and Echo-State Networks for Predicting Potential Railway Door System Failures." In: *IEEE Transactions on Reliability* 64.3 (Sept. 2015), pp. 861–868. ISSN: 1558-1721. DOI: 10.1109/TR.2015.2424213.

[50] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. "Unsupervised Scalable Representation Learning for Multivariate Time Series." In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019, p. 12.

[51] J. F. G. de Freitas et al. "Sequential Monte Carlo Methods to Train Neural Network Models." In: *Neural Computation* 12.4 (Apr. 2000), pp. 955–993. ISSN: 0899-7667. DOI: 10.1162/089976600300015664.

[52] Roger Frigola. "Bayesian Time Series Learning with Gaussian Processes." PhD thesis. University of Cambridge, Aug. 2015.

[53] Song Fu et al. "A Novel Time-Series Memory Auto-Encoder With Sequentially Updated Reconstructions for Remaining Useful Life Prediction." In: *IEEE Transactions on Neural Networks and Learning Systems* PP (June 2021), pp. 1–12. DOI: 10.1109/TNNLS.2021.3084249.

[54] Nagi Gebraeel. "Sensory-Updated Residual Life Distributions for Components With Exponential Degradation Patterns." In: *Automation Science and Engineering, IEEE Transactions on* 3 (Nov. 2006), pp. 382–393. DOI: 10.1109/TASE.2006.876609.

[55] Zoubin Ghahramani. "Unsupervised Learning." In: *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*. Ed. by Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004, pp. 72–112. ISBN: 978-3-540-28650-9. DOI: 10.1007/978-3-540-28650-9_5.

[56] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Nov. 2016. ISBN: 978-0-262-33737-3.

[57] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. "Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation." In: *IEE Proceedings F (Radar and Signal Processing)* 140.2 (Apr. 1993), pp. 107–113. ISSN: 2053-9045. DOI: 10.1049/ip-f-2.1993.0015.

[58] Henry Gouk et al. "Regularisation of Neural Networks by Enforcing Lipschitz Continuity." In: *Machine Learning* 110.2 (Feb. 2021), pp. 393–416. ISSN: 1573-0565. DOI: 10.1007/s10994-020-05929-w.

[59] Mikhail Gromov. *Metric Structures for Riemannian and Non-Riemannian Spaces*. Modern Birkhäuser Classics. Boston, MA: Birkhäuser, 2007. ISBN: 978-0-8176-4582-3. DOI: 10.1007/978-0-8176-4583-0.

[60] Narendhar Gugulothu et al. "Predicting Remaining Useful Life Using Time Series Embeddings Based on Recurrent Neural Networks." In: *arXiv:1709.01073 [cs]* (Oct. 2017). arXiv: 1709.01073 [cs].

[61] Jie Gui et al. *A Survey of Self-supervised Learning from Multiple Perspectives: Algorithms, Applications and Future Trends*. Aug. 2023. DOI: 10.48550/arXiv.2301.05712. arXiv: 2301.05712 [cs].

[62] Liang Guo et al. "A Recurrent Neural Network Based Health Indicator for Remaining Useful Life Prediction of Bearings." In: *Neurocomputing* 240 (May 2017), pp. 98–109. ISSN: 09252312. DOI: 10.1016/j.neucom.2017.02.045.

[63] Ashish Gupta et al. "An Unseen Fault Classification Approach for Smart Appliances Using Ongoing Multivariate Time Series." In: *IEEE Transactions on Industrial Informatics* 17.6 (June 2021), pp. 3731–3738. ISSN: 1941-0050. DOI: 10.1109/TII.2020.3016590.

[64] Mai Lan Ha and Volker Blanz. "Deep Ranking with Adaptive Margin Triplet Loss." In: *arXiv:2107.06187 [cs]* (July 2021). arXiv: 2107.06187 [cs].

[65] R. Hadsell, S. Chopra, and Y. LeCun. "Dimensionality Reduction by Learning an Invariant Mapping." In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. June 2006, pp. 1735–1742. DOI: 10.1109/CVPR.2006.100.

[66] David Heckerman. "A Tutorial on Learning with Bayesian Networks." In: *Innovations in Bayesian Networks: Theory and Applications*. Ed. by Dawn E. Holmes and Lakhmi C. Jain. Studies in Computational Intelligence. Berlin, Heidelberg: Springer, 2008, pp. 33–82. ISBN: 978-3-540-85066-3. DOI: 10.1007/978-3-540-85066-3_3.

[67] Ahmed Zakariae Hinchi and Mohamed Tkiouat. "Rolling Element Bearing Remaining Useful Life Estimation Based on a Convolutional Long-Short-Term Memory Network." In: *Procedia Computer Science*. PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON INTELLIGENT COMPUTING IN DATA SCIENCES, ICDS2017 127 (Jan. 2018), pp. 123–132. ISSN: 1877-0509. DOI: 10.1016/j.procs.2018.01.106.

[68] C. Hu et al. "A Prognostic Model Based on DBN and Diffusion Process for Degrading Bearing." In: *IEEE Transactions on Industrial Electronics* 67.10 (Oct. 2020), pp. 8767–8777. ISSN: 1557-9948. DOI: 10.1109/TIE.2019.2947839.

[69] Xiaoxi Hu et al. "Data-Driven Technology of Fault Diagnosis in Railway Point Machines: Review and Challenges." In: *Transportation Safety and Environment* 4.4 (Dec. 2022), tdac036. ISSN: 2631-4428. DOI: 10.1093/tse/tdac036.

[70] Cheng Hua et al. "Performance Reliability Estimation Method Based on Adaptive Failure Threshold." In: *Mechanical Systems and Signal Processing* 36.2 (Apr. 2013), pp. 505–519. ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2012.10.019.

[71] Cheng-Geng Huang et al. "A Novel Deep Convolutional Neural Network-Bootstrap Integrated Method for RUL Prediction of Rolling Bearing." In: *Journal of Manufacturing Systems* (Mar. 2021). ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2021.03.012.

[72] Henry Jackson. "Race, Economy and Punishment: Inequity and Racial Disparity in Imprisonment, 1972–2002." In: *Criminal Justice Studies* 27.2 (Apr. 2014), pp. 226–243. ISSN: 1478-601X. DOI: 10.1080/1478601X.2013.870073.

[73] Andrew R. Jacobson et al. "Climate Forcing and Density Dependence in a Mountain Ungulate Population." In: *Ecology* 85.6 (2004), pp. 1598–1610. ISSN: 1939-9170. DOI: 10.1890/02-0753.

[74] Tim Januschowski et al. "Forecasting with Trees." In: *International Journal of Forecasting*. Special Issue: M5 Competition 38.4 (Oct. 2022), pp. 1473–1481. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2021.10.004.

[75] Kamran Javed, Rafael Gouriveau, and Noureddine Zerhouni. "Novel Failure Prognostics Approach with Dynamic Thresholds for Machine Degradation." In: *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*. Vienna, Austria: IEEE, Nov. 2013, pp. 4404–4409. ISBN: 978-1-4799-0224-8. DOI: 10.1109/IECON.2013.6699844.

[76] Ranjeet Ranjan Jha et al. "HLGSNet: Hierarchical and Lightweight Graph Siamese Network with Triplet Loss for fMRI-based Classification of ADHD." In: *2020 International Joint Conference on Neural Networks (IJCNN)*. July 2020, pp. 1–7. DOI: 10.1109/IJCNN48605.2020.9207667.

[77] Jianfang Jia et al. "SOH and RUL Prediction of Lithium-Ion Batteries Based on Gaussian Process Regression with Indirect Health Indicators." In: *Energies* 13.2 (Jan. 2020), p. 375. DOI: 10.3390/en13020375.

[78] Sun Jianzhong et al. "Study of Ensemble Learning-Based Fusion Prognostics." In: *2010 Prognostics and System Health Management Conference*. Jan. 2010, pp. 1–7. DOI: 10.1109/PHM.2010.5414582.

[79] Ruihua Jiao et al. "Fault Monitoring and Remaining Useful Life Prediction Framework for Multiple Fault Modes in Prognostics." In: *Reliability Engineering & System Safety* 203 (Nov. 2020), p. 107028. ISSN: 0951-8320. DOI: 10.1016/j.ress.2020.107028.

[80] Michael L. Johnson and Lindsay M. Faunt. "[1] Parameter Estimation by Least-Squares Methods." In: *Methods in Enzymology*. Vol. 210. Numerical Computer Methods. Academic Press, Jan. 1992, pp. 1–37. DOI: 10.1016/0076-6879(92)10003-V.

[81] Santosh Kabbur, Xia Ning, and George Karypis. "FISM: Factored Item Similarity Models for Top-N Recommender Systems." In: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '13. New York, NY, USA: Association for Computing Machinery, Aug. 2013, pp. 659–667. ISBN: 978-1-4503-2174-7. DOI: 10.1145/2487575.2487589.

[82] Zhao Kang, Chong Peng, and Qiang Cheng. "Kernel-Driven Similarity Learning." In: *Neurocomputing* 267 (Dec. 2017), pp. 210–219. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.06.005.

[83] Nikolas Kantas et al. "On Particle Methods for Parameter Estimation in State-Space Models." In: *Statistical Science* 30.3 (Aug. 2015). ISSN: 0883-4237. DOI: 10.1214/14-STS511. arXiv: 1412.8695 [stat].

[84] Guolin Ke et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree." In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[85] J. Kennedy and R. Eberhart. "Particle Swarm Optimization." In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. Nov. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[86] Abbas Khosravi et al. "Comprehensive Review of Neural Network-Based Prediction Intervals and New Advances." In: *IEEE Transactions on Neural Networks* 22.9 (Sept. 2011), pp. 1341–1356. ISSN: 1941-0093. DOI: 10.1109/TNN.2011.2162110.

[87] Phattara Khumprom and Nita Yodo. "A Data-Driven Predictive Prognostic Model for Lithium-ion Batteries Based on a Deep Learning Algorithm." In: *Energies* 12.4 (Jan. 2019), p. 660. DOI: 10.3390/en12040660.

[88] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 2017. DOI: 10.48550/arXiv.1412.6980. arXiv: 1412.6980 [cs].

[89] Alex Kosgodagan-Dalla Torre et al. "Representing Markov processes as dynamic non-parametric Bayesian networks." In: *HAL* 2017.0 (2017).

[90] S. B. Kotsiantis. "Decision Trees: A Recent Overview." In: *Artificial Intelligence Review* 39.4 (Apr. 2013), pp. 261–283. ISSN: 1573-7462. DOI: 10.1007/s10462-011-9272-4.

[91] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012.

[92] Brian Kulis. "Metric Learning: A Survey." In: *Foundations and Trends® in Machine Learning* 5.4 (July 2013), pp. 287–364. ISSN: 1935-8237, 1935-8245. DOI: 10.1561/2200000019.

[93] Anil Kumar et al. "State-Space Modeling and Novel Entropy-Based Health Indicator for Dynamic Degradation Monitoring of Rolling Element Bearing." In: *Reliability Engineering & System Safety* 221 (May 2022), p. 108356. ISSN: 0951-8320. DOI: 10.1016/j.ress.2022.108356.

[94] Ashutosh Kumar Dubey et al. "Study and Analysis of SARIMA and LSTM in Forecasting Time Series Data." In: *Sustainable Energy Technologies and Assessments* 47 (Oct. 2021), p. 101474. ISSN: 2213-1388. DOI: 10.1016/j.seta.2021.101474.

[95] Divina Lawrance and Suja Palaniswamy. "Emotion Recognition from Facial Expressions for 3D Videos Using Siamese Network." In: *2021 International Conference on Communication, Control and Information Sciences (ICCISc)*. Vol. 1. June 2021, pp. 1–6. DOI: 10.1109/ICCISc52257.2021.9484949.

[96] Jay Lee et al. "Prognostics and Health Management Design for Rotary Machinery Systems—Reviews, Methodology and Applications." In: *Mechanical Systems and Signal Processing* 42.1 (Jan. 2014), pp. 314–334. ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2013.06.004.

[97] Yaguo Lei et al. "A Model-Based Method for Remaining Useful Life Prediction of Machinery." In: *IEEE Transactions on Reliability* 65.3 (Sept. 2016), pp. 1314–1326. ISSN: 0018-9529, 1558-1721. DOI: 10.1109/TR.2016.2570568.

[98] Yaguo Lei et al. "Machinery Health Prognostics: A Systematic Review from Data Acquisition to RUL Prediction." In: *Mechanical Systems and Signal Processing* 104 (May 2018), pp. 799–834. ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2017.11.016.

[99] Hongru Li et al. "A Novel Method Based upon Modified Composite Spectrum and Relative Entropy for Degradation Feature Extraction of Hydraulic Pump." In: *Mechanical Systems and Signal Processing* 114 (Jan. 2019), pp. 399–412. ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2018.04.040.

[100] Hua Li et al. "Application of EEMD and Improved Frequency Band Entropy in Bearing Fault Feature Extraction." In: *ISA Transactions* 88 (May 2019), pp. 170–185. ISSN: 0019-0578. DOI: 10.1016/j.isatra.2018.12.002.

[101] Naipeng Li et al. "A Wiener-Process-Model-Based Method for Remaining Useful Life Prediction Considering Unit-to-Unit Variability." In: *IEEE Transactions on Industrial Electronics* 66.3 (Mar. 2019), pp. 2092–2101. ISSN: 0278-0046, 1557-9948. DOI: 10.1109/TIE.2018.2838078.

[102] Naipeng Li et al. "An Improved Exponential Model for Predicting Remaining Useful Life of Rolling Element Bearings." In: *IEEE Transactions on Industrial Electronics* 62.12 (Dec. 2015), pp. 7762–7773. ISSN: 0278-0046, 1557-9948. DOI: 10.1109/TIE.2015.2455055.

[103] Penghua Li et al. "State-of-Health Estimation and Remaining Useful Life Prediction for the Lithium-Ion Battery Based on a Variant Long Short Term Memory Neural Network." In: *Journal of Power Sources* 459 (May 2020), p. 228069. ISSN: 0378-7753. DOI: 10.1016/j.jpowsour.2020.228069.

[104] Ting Li, Shuguang He, and Xiujie Zhao. "Optimal Warranty Policy Design for Deteriorating Products with Random Failure Threshold." In: *Reliability Engineering & System Safety* 218 (Feb. 2022), p. 108142. ISSN: 0951-8320. DOI: 10.1016/j.ress.2021.108142.

[105] Xiang Li, Wei Zhang, and Qian Ding. "Deep Learning-Based Remaining Useful Life Estimation of Bearings Using Multi-Scale Feature Extraction." In: *Reliability Engineering & System Safety* 182 (Feb. 2019), pp. 208–218. ISSN: 0951-8320. DOI: 10.1016/j.ress.2018.11.011.

[106] Xin Li et al. "Tipping Point Detection Using Reservoir Computing." In: *Research* 6 (July 2023), p. 0174. DOI: 10.34133/research.0174.

[107] L. Liao and F. Köttig. "Review of Hybrid Prognostics Approaches for Remaining Useful Life Prediction of Engineered Systems, and an Application to Battery Life Prediction." In: *IEEE Transactions on Reliability* 63.1 (Mar. 2014), pp. 191–207. ISSN: 1558-1721. DOI: 10.1109/TR.2014.2299152.

[108] Linxia Liao, Wenjing Jin, and Radu Pavel. "Enhanced Restricted Boltzmann Machine With Prognosability Regularization for Prognostics and Health Assessment." In: *IEEE Transactions on Industrial Electronics* 63.11 (Nov. 2016), pp. 7076–7083. ISSN: 1557-9948. DOI: 10.1109/TIE.2016.2586442.

[109] Bryan Lim et al. "Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting." In: *arXiv:1912.09363 [cs, stat]* (Sept. 2020). arXiv: 1912.09363 [cs, stat].

[110] D. Liu et al. "A Health Indicator Extraction and Optimization Framework for Lithium-Ion Battery Degradation Modeling and Prognostics." In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.6 (June 2015), pp. 915–928. ISSN: 2168-2232. DOI: 10.1109/TSMC.2015.2389757.

[111] Kaibo Liu, Abdallah Chehade, and Changyue Song. "Optimize the Signal Quality of the Composite Health Index via Data Fusion for Degradation Modeling and Prognostic Analysis." In: *IEEE Transactions on Automation Science and Engineering* 14.3 (July 2017), pp. 1504–1514. ISSN: 1545-5955, 1558-3783. DOI: 10.1109/TASE.2015.2446752.

[112] Mengsi Liu et al. "Mixed Similarity Learning for Recommendation with Implicit Feedback." In: *Knowledge-Based Systems* 119 (Mar. 2017), pp. 178–185. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2016.12.010.

[113] Yingchao Liu, Xiaofeng Hu, and Wenjuan Zhang. "Remaining Useful Life Prediction Based on Health Index Similarity." In: *Reliability Engineering & System Safety* 185 (May 2019), pp. 502–510. ISSN: 0951-8320. DOI: 10.1016/j.ress.2019.02.002.

[114] C. Joseph Lu and William O. Meeker. "Using Degradation Measures to Estimate a Time-to-Failure Distribution." In: *Technometrics* 35.2 (May 1993), pp. 161–174. ISSN: 0040-1706. DOI: 10.1080/00401706.1993.10485038.

[115] David Luengo et al. "A Survey of Monte Carlo Methods for Parameter Estimation." In: *EURASIP Journal on Advances in Signal Processing* 2020.1 (May 2020), p. 25. ISSN: 1687-6180. DOI: 10.1186/s13634-020-00675-6.

[116] Helmut Lütkepohl. "Chapter 6 Forecasting with VARMA Models." In: *Handbook of Economic Forecasting*. Ed. by G. Elliott, C. W. J. Granger, and A. Timmermann. Vol. 1. Elsevier, Jan. 2006, pp. 287–325. DOI: 10.1016/S1574-0706(05)01006-2.

[117] Biao Ma et al. "Similarity-Based Failure Threshold Determination for System Residual Life Prediction." In: *Eksploatacja i Niezawodność* Vol. 22.no. 3 (2020). ISSN: 1507-2711. DOI: 10.17531/ein.2020.3.15.

[118] Walter Zucchini MacDonald Iain L. *Hidden Markov Models for Time Series: An Introduction Using R*. New York: Chapman and Hall/CRC, Apr. 2009. ISBN: 978-0-429-13953-6. DOI: 10.1201/9781420010893.

[119] James A. Macinko, Leiyu Shi, and Barbara Starfield. "Wage Inequality, the Health System, and Infant Mortality in Wealthy Industrialized Countries, 1970–1996." In: *Social Science & Medicine*. Adjusting for Market Failure: Challenges in Public Health Alternatives 58.2 (Jan. 2004), pp. 279–292. ISSN: 0277-9536. DOI: 10.1016/S0277-9536(03)00200-4.

[120] Chu V. Mai et al. "Surrogate Modeling for Stochastic Dynamical Systems by Combining Nonlinear Autoregressive with Exogenous Input Models and Polynomial Chaos Expansions." In: *International Journal for Uncertainty Quantification* 6.4 (2016). ISSN: 2152-5080, 2152-5099. DOI: 10.1615/Int.J.UncertaintyQuantification.2016016603.

[121] Pankaj Malhotra et al. *Multi-Sensor Prognostics Using an Unsupervised Health Index Based on LSTM Encoder-Decoder*. Aug. 2016. arXiv: 1608.06154 [cs].

[122] M. M. Manjurul Islam, Alexander E. Prosvirin, and Jong-Myon Kim. "Data-Driven Prognostic Scheme for Rolling-Element Bearings Using a New Health Index and Variants of Least-Square Support Vector Machines." In: *Mechanical Systems and Signal Processing* 160 (Nov. 2021), p. 107853. ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2021.107853.

[123] Timothy Masters. *Practical Neural Network Recipes in C++*. USA: Academic Press Professional, Inc., 1993. ISBN: 978-0-12-479040-7.

[124] K. Medjaher, N. Zerhouni, and J. Baklouti. "Data-Driven Prognostics Based on Health Indicator Construction: Application to PRONOSTIA's Data." In: *2013 European Control Conference (ECC)*. Zurich: IEEE, July 2013, pp. 1451–1456. ISBN: 978-3-033-03962-9. DOI: 10.23919/ECC.2013.6669223.

[125] Safyan Aman Memon, Kinaan Aamir Khan, and Hammad Naveed. "HECNet: A Hierarchical Approach to Enzyme Function Classification Using a Siamese Triplet Network." In: *Bioinformatics* 36.17 (Nov. 2020), pp. 4583–4589. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btaa536.

[126] A Mignatti, G Corani, and Andrea-Emilio Rizzoli. "Credal Model Averaging: Dealing Robustly with Model Uncertainty on Small Data Sets." In: (), p. 9.

[127] Seyedali Mirjalili. "Genetic Algorithm." In: *Evolutionary Algorithms and Neural Networks: Theory and Applications*. Ed. by Seyedali Mirjalili. Studies in Computational Intelligence. Cham: Springer International Publishing, 2019, pp. 43–55. ISBN: 978-3-319-93025-1. DOI: 10.1007/978-3-319-93025-1_4.

[128] Francesco Carlo Morabito et al. "Chapter 11 - Deep Learning Approaches to Electrophysiological Multivariate Time-Series Analysis$_{**}$To My Loved Daughter, Valeria." In: *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Ed. by Robert Kozma et al. Academic Press, Jan. 2019, pp. 219–243. ISBN: 978-0-12-815480-9. DOI: 10.1016/B978-0-12-815480-9.00011-6.

[129] Jorge J. Moré. "The Levenberg-Marquardt Algorithm: Implementation and Theory." In: *Numerical Analysis*. Ed. by G. A. Watson. Vol. 630. Berlin, Heidelberg: Springer Berlin Heidelberg, 1978, pp. 105–116. ISBN: 978-3-540-08538-6 978-3-540-35972-2. DOI: 10.1007/BFb0067700.

[130] Mohammad Amin Morid et al. "Learning Hidden Patterns from Patient Multivariate Time Series Data Using Convolutional Neural Networks: A Case Study of Healthcare Cost Prediction." In: *Journal of Biomedical Informatics* 111 (Nov. 2020), p. 103565. ISSN: 1532-0464. DOI: 10.1016/j.jbi.2020.103565.

[131] A. Mosallam, K. Medjaher, and N. Zerhouni. "Data-Driven Prognostic Method Based on Bayesian Approaches for Direct Remaining Useful Life Prediction." In: *Journal of Intelligent Manufacturing* 27.5 (Oct. 2016), pp. 1037–1048. ISSN: 0956-5515, 1572-8145. DOI: 10.1007/s10845-014-0933-4.

[132] Robert Moskovitch and Yuval Shahar. "Classification-Driven Temporal Discretization of Multivariate Time Series." In: *Data Mining and Knowledge Discovery* 29.4 (July 2015), pp. 871–913. ISSN: 1573-756X. DOI: 10.1007/s10618-014-0380-z.

[133] Kevin Patrick Murphy. "Dynamic Bayesian Networks: Representation, Inference and Learning." PhD thesis. 2002.

[134] Fionn Murtagh, Michael Spagat, and Jorge A. Restrepo. "Ultrametric Wavelet Regression of Multivariate Time Series: Application to Colombian Conflict Analysis." In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 41.2 (Mar. 2011), pp. 254–263. ISSN: 1558-2426. DOI: 10.1109/TSMCA.2010.2064301.

[135] Patrick Nectoux et al. "PRONOSTIA : An Experimental Platform for Bearings Accelerated Degradation Tests." In: *IEEE International Conference on Prognostics and Health Management, PHM'12.* Vol. sur CD ROM. IEEE Catalog Number : CPF12PHM-CDR, June 2012, p. 1.

[136] Arkadi S. Nemirovski and Michael J. Todd. "Interior-Point Methods for Optimization." In: *Acta Numerica* 17 (May 2008), pp. 191–234. ISSN: 1474-0508, 0962-4929. DOI: 10.1017/S0962492906370018.

[137] Paul Newbold. "ARIMA Model Building and the Time Series Analysis Approach to Forecasting." In: *Journal of Forecasting* 2.1 (1983), pp. 23–35. ISSN: 1099-131X. DOI: 10.1002/for.3980020104.

[138] Khanh T. P. Nguyen and Kamal Medjaher. "An Automated Health Indicator Construction Methodology for Prognostics Based on Multi-Criteria Optimization." In: *ISA Transactions* 113 (July 2021), pp. 81–96. ISSN: 0019-0578. DOI: 10.1016/j.isatra.2020.03.017.

[139] Bent Helge Nystad, Giulio Gola, and John Einar Hulsund. "Lifetime Models for Remaining Useful Life Estimation with Randomly Distributed Failure Thresholds." In: *PHM Society European Conference* 1.1 (2012). ISSN: 2325-016X. DOI: 10.36001/phme.2012.v1i1.1442.

[140] Hideki Oki et al. "Triplet Loss for Knowledge Distillation." In: *2020 International Joint Conference on Neural Networks (IJCNN)*. July 2020, pp. 1–7. DOI: 10.1109/IJCNN48605.2020.9207148.

[141] Charles H. Oppenheimer and Kenneth A. Loparo. "Physically Based Diagnosis and Prognosis of Cracked Rotor Shafts." In: *Component and Systems Diagnostics, Prognostics, and Health Management II*. Vol. 4733. SPIE, July 2002, pp. 122–132. DOI: 10.1117/12.475502.

[142] Murphy K. P. "Dynamic Bayesian Networks : Representation, Inference and Learning, Dissertation." In: *PhD thesis, U.C. Berkley, Dept. Comp. Sci* (2002).

[143] Giovanni Pagliarini et al. "Neural-Symbolic Temporal Decision Trees for Multivariate Time Series Classification." In: *29th International Symposium on Temporal Representation and Reasoning (TIME 2022)*. Ed. by Alexander Artikis, Roberto Posenato, and Stefano Tonetta. Vol. 247. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 13:1–13:15. ISBN: 978-3-95977-262-4. DOI: 10.4230/LIPIcs.TIME.2022.13.

[144] Joan Pellegrino et al. *Measurement Science Roadmap for Prognostics and Health Management for Smart Manufacturing Systems*. Tech. rep. NIST AMS 100-2. National Institute of Standards and Technology, Sept. 2016, NIST AMS 100–2. DOI: 10.6028/NIST.AMS.100-2.

[145] Kaixiang Peng et al. "A Deep Belief Network Based Health Indicator Construction and Remaining Useful Life Prediction Using Improved Particle Filter." In: *Neurocomputing* 361 (Oct. 2019), pp. 19–28. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2019.07.075.

[146] A. Picot et al. "Statistic-Based Spectral Indicator for Bearing Fault Detection in Permanent-Magnet Synchronous Machines Using the Stator Current." In: *Mechanical Systems and Signal Processing* 46.2 (June 2014), pp. 424–441. ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2014.01.006.

[147] M. Andrea Previtali et al. "Population Dynamics of Two Sympatric Rodents in a Variable Environment: Rainfall, Resource Availability, and Predation." In: *Ecology* 90.7 (2009), pp. 1996–2006. ISSN: 1939-9170. DOI: 10.1890/08-0405.1.

[148] Fang Qian and Gang Niu. "Remaining Useful Life Prediction Using Ranking Mutual Information Based Monotonic Health Indicator." In: *2015 Prognostics and System Health Management Conference (PHM)*. Oct. 2015, pp. 1–5. DOI: 10.1109/PHM.2015.7380042.

[149] Yuning Qian, Ruqiang Yan, and Robert X. Gao. "A Multi-Time Scale Approach to Remaining Useful Life Prediction in Rolling Bearing." In: *Mechanical Systems and Signal Processing* 83 (Jan. 2017), pp. 549–567. ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2016.06.031.

[150] Yi Qin, Jianghong Zhou, and Dingliang Chen. "Unsupervised Health Indicator Construction by a Novel Degradation-Trend-Constrained Variational Autoencoder and Its Applications." In: *IEEE/ASME Transactions on Mechatronics* 27.3 (June 2022), pp. 1447–1456. ISSN: 1083-4435, 1941-014X. DOI: 10.1109/TMECH.2021.3098737.

[151] Yi Qin et al. "Gated Dual Attention Unit Neural Networks for Remaining Useful Life Prediction of Rolling Bearings." In: *IEEE Transactions on Industrial Informatics* 17.9 (Sept. 2021), pp. 6438–6447. ISSN: 1941-0050. DOI: 10.1109/TII.2020.2999442.

[152] Guangqi Qiu, Yingkui Gu, and Junjie Chen. "Selective Health Indicator for Bearings Ensemble Remaining Useful Life Prediction with Genetic Algorithm and Weibull Proportional Hazards Model." In: *Measurement* 150 (Jan. 2020), p. 107097. ISSN: 0263-2241. DOI: 10.1016/j.measurement.2019.107097.

[153] Akhand Rai and Jong-Myon Kim. "A Novel Health Indicator Based on the Lyapunov Exponent, a Probabilistic Self-Organizing Map, and the Gini-Simpson Index for Calculating the RUL of Bearings." In: *Measurement* 164 (Nov. 2020), p. 108002. ISSN: 0263-2241. DOI: 10.1016/j.measurement.2020.108002.

[154] Emmanuel Ramasso. "Investigating Computational Geometry for Failure Prognostics." In: *International Journal of Prognostics and Health Management* 5.1 (Nov. 2020). ISSN: 2153-2648, 2153-2648. DOI: 10.36001/ijphm.2014.v5i1.2205.

[155] Saeed Ramezani, Alireza Moini, and Mohamad Riahi. "Prognostics and Health Management in Machinery: A Review of Methodologies for RUL Prediction and Roadmap." In: *International Journal of Industrial Engineering and Management Science* 6.1 (Apr. 2019), pp. 38–61. ISSN: 2409-1871.

[156] Syama Sundar Rangapuram et al. "Deep State Space Models for Time Series Forecasting." In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018.

[157] Friedrich Recknagel et al. "Comparative Application of Artificial Neural Networks and Genetic Algorithms for Multivariate Time-Series Modelling of Algal Blooms in Freshwater Lakes." In: *Journal of Hydroinformatics* 4.2 (Mar. 2002), pp. 125–133. ISSN: 1464-7141. DOI: 10.2166/hydro.2002.0013.

[158] Fuji Ren and Siyuan Xue. "Intention Detection Based on Siamese Neural Network With Triplet Loss." In: *IEEE Access* 8 (2020), pp. 82242–82254. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2991484.

[159] Behnoush Rezaeianjouybari and Yi Shang. "Deep Learning for Prognostics and Health Management: State of the Art, Challenges, and Opportunities." In: *Measurement* 163 (Oct. 2020), p. 107929. ISSN: 0263-2241. DOI: 10.1016/j.measurement.2020.107929.

[160] Marco Rigamonti et al. "Ensemble of Optimized Echo State Networks for Remaining Useful Life Prediction." In: *Neurocomputing* 281 (Mar. 2018), pp. 121–138. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.11.062.

[161] S. Roberts et al. "Gaussian Processes for Time-Series Modelling." In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371.1984 (Feb. 2013), p. 20110550. DOI: 10.1098/rsta.2011.0550.

[162] Lior Rokach and Oded Maimon. "Clustering Methods." In: *Data Mining and Knowledge Discovery Handbook*. Ed. by Oded Maimon and Lior Rokach. Boston, MA: Springer US, 2005, pp. 321–352. ISBN: 978-0-387-25465-4. DOI: 10.1007/0-387-25465-X_15.

[163] Peter M. Roth et al. "Mahalanobis Distance Learning for Person Re-identification." In: *Person Re-Identification*. Ed. by Shaogang Gong et al. Advances in Computer Vision and Pattern Recognition. London: Springer, 2014, pp. 247–267. ISBN: 978-1-4471-6296-4. DOI: 10.1007/978-1-4471-6296-4_12.

[164] Sebastian Ruder. *An Overview of Gradient Descent Optimization Algorithms*. June 2017. DOI: 10.48550/arXiv.1609.04747. arXiv: 1609.04747 [cs].

[165] Omer Sagi and Lior Rokach. "Ensemble Learning: A Survey." In: *WIREs Data Mining and Knowledge Discovery* 8.4 (2018), e1249. ISSN: 1942-4795. DOI: 10.1002/widm.1249.

[166] Abhinav Saxena and Kai Goebel. "Turbofan Engine Degradation Simulation Data Set." In: *NASA Ames Prognostics Data Repository* (2008), pp. 1551–3203.

[167] Abhinav Saxena, Don Simon, and Neil Eklund. "Damage Propagation Modeling for Aircraft Engine Prognostics." In: (), p. 9.

[168] Robert E. Schapire. "Explaining AdaBoost." In: *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*. Ed. by Bernhard Schölkopf, Zhiyuan Luo, and Vladimir Vovk. Berlin, Heidelberg: Springer, 2013, pp. 37–52. ISBN: 978-3-642-41136-6. DOI: 10.1007/978-3-642-41136-6_5.

[169] Bernhard Schölkopf. "The Kernel Trick for Distances." In: ().

[170] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A Unified Embedding for Face Recognition and Clustering." In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015), pp. 815–823. DOI: 10.1109/CVPR.2015.7298682. arXiv: 1503.03832.

[171] Guido Sciavicco and Stan Ionel Eduard. *Knowledge Extraction with Interval Temporal Logic Decision Trees*. May 2023. DOI: 10.48550/arXiv.2305.16864. arXiv: 2305.16864 [cs].

[172] David W. Scott. "Multivariate Density Estimation and Visualization." In: *Handbook of Computational Statistics: Concepts and Methods*. Ed. by James E. Gentle, Wolfgang Karl Härdle, and Yuichi Mori. Springer Handbooks of Computational Statistics. Berlin, Heidelberg: Springer, 2012, pp. 549–569. ISBN: 978-3-642-21551-3. DOI: 10.1007/978-3-642-21551-3_19.

[173] Matthias Seeger. "Gaussian Processes for Machine Learning." In: *International Journal of Neural Systems* 14.02 (Apr. 2004), pp. 69–106. ISSN: 0129-0657. DOI: 10.1142/S0129065704001899.

[174] N. Shephard and T. Flury. "Learning and Filtering via Simulation: Smoothly Jittered Particle Filters." In: (2009).

[175] Xiao-Sheng Si et al. "Remaining Useful Life Estimation – A Review on the Statistical Data Driven Approaches." In: *European Journal of Operational Research* 213.1 (Aug. 2011), pp. 1–14. ISSN: 03772217. DOI: 10.1016/j.ejor.2010.11.018.

[176] Xiao-Sheng Si et al. "Remaining Useful Life Estimation Based on a Nonlinear Diffusion Degradation Process." In: *IEEE Transactions on Reliability* 1.61 (2012), pp. 50–67. ISSN: 0018-9529, 1558-1721. DOI: 10.1109/TR.2011.2182221.

[177] Edward A. Silver et al. "A Tutorial on Heuristic Methods." In: *European Journal of Operational Research* 5.3 (Sept. 1980), pp. 153–162. ISSN: 0377-2217. DOI: 10.1016/0377-2217(80)90084-3.

[178] Ashish Singhal and Dale E. Seborg. "Clustering Multivariate Time-Series Data." In: *Journal of Chemometrics* 19.8 (2005), pp. 427–438. ISSN: 1099-128X. DOI: 10.1002/cem.945.

[179] Tanin Sirimongkolkasem and Reza Drikvandi. "On Regularisation Methods for Analysis of High Dimensional Data." In: *Annals of Data Science* 6.4 (Dec. 2019), pp. 737–763. ISSN: 2198-5812. DOI: 10.1007/s40745-019-00209-4.

[180] Changyue Song, Kaibo Liu, and Xi Zhang. "Integration of Data-Level Fusion Model and Kernel Methods for Degradation Modeling and Prognostic Analysis." In: *IEEE Transactions on Reliability* 67.2 (June 2018), pp. 640–650. ISSN: 0018-9529, 1558-1721. DOI: 10.1109/TR.2017.2715180.

[181] Abdenour Soualhi, Kamal Medjaher, and Noureddine Zerhouni. "Bearing Health Monitoring Based on Hilbert–Huang Transform, Support Vector Machine, and Regression." In: *IEEE Transactions on Instrumentation and Measurement* 64.1 (Jan. 2015), pp. 52–62. ISSN: 0018-9456, 1557-9662. DOI: 10.1109/TIM.2014.2330494.

[182] Ruiqi Sun et al. "Dynamic Forecast of Desert Locust Presence Using Machine Learning with a Multivariate Time Lag Sliding Window Technique." In: *Remote Sensing* 14.3 (Jan. 2022), p. 747. ISSN: 2072-4292. DOI: 10.3390/rs14030747.

[183] Shan Suthaharan. "Support Vector Machine." In: *Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning*. Ed. by Shan Suthaharan. Integrated Series in Information Systems. Boston, MA: Springer US, 2016, pp. 207–235. ISBN: 978-1-4899-7641-3. DOI: 10.1007/978-1-4899-7641-3_9.

[184] Thamo Sutharssan et al. "Prognostic and Health Management for Engineering Systems: A Review of the Data-Driven Approach and Algorithms." In: *The Journal of Engineering* 2015.7 (2015), pp. 215–222. ISSN: 2051-3305. DOI: 10.1049/joe.2014.0303.

[185] Benjamin Szubert et al. "Structure-Preserving Visualisation of High Dimensional Single-Cell Datasets." In: *Scientific Reports* 9.1 (June 2019), p. 8914. ISSN: 2045-2322. DOI: 10.1038/s41598-019-45301-0.

[186] Chang Wei Tan et al. "Time Series Extrinsic Regression." In: *Data Mining and Knowledge Discovery* 35.3 (May 2021), pp. 1032–1060. ISSN: 1573-756X. DOI: 10.1007/s10618-021-00745-9.

[187] Fengzhen Tang. "KERNEL METHODS FOR TIME SERIES DATA." PhD thesis. July 2015.

[188] Sheng-jin Tang et al. "Remaining Useful Life Estimation Based on Wiener Degradation Processes with Random Failure Threshold." In: *Journal of Central South University* 23.9 (Sept. 2016), pp. 2230–2241. ISSN: 2227-5223. DOI: 10.1007/s11771-016-3281-z.

[189] Shengjin Tang et al. "Remaining Useful Life Prediction of Lithium-Ion Batteries Based on the Wiener Process with Measurement Error." In: *Energies* 7.2 (Feb. 2014), pp. 520–547. DOI: 10.3390/en7020520.

[190] Andreas Theissler et al. "Predictive Maintenance Enabled by Machine Learning: Use Cases and Challenges in the Automotive Industry." In: *Reliability Engineering & System Safety* 215 (Nov. 2021), p. 107864. ISSN: 0951-8320. DOI: 10.1016/j.ress.2021.107864.

[191] Christopher Torrence and Gilbert P Compo. "A Practical Guide to Wavelet Analysis." In: *Bulletin of the American Meteorological Society* 79.1 (1998), p. 18.

[192] Yiu L. Tse, Michael E. Cholette, and Peter W. Tse. "A Multi-Sensor Approach to Remaining Useful Life Estimation for a Slurry Pump." In: *Measurement* 139 (June 2019), pp. 140–151. ISSN: 02632241. DOI: 10.1016/j.measurement.2019.02.079.

[193] Julio J. Valdés and Graeme Bonham-Carter. "Time Dependent Neural Network Models for Detecting Changes of State in Complex Processes: Applications in Earth Sciences and Astronomy." In: *Neural Networks*. Earth Sciences and Environmental Applications of Computational Intelligence 19.2 (Mar. 2006), pp. 196–207. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2006.01.006.

[194] Jesper E. van Engelen and Holger H. Hoos. "A Survey on Semi-Supervised Learning." In: *Machine Learning* 109.2 (Feb. 2020), pp. 373–440. ISSN: 1573-0565. DOI: 10.1007/s10994-019-05855-6.

[195] Ashish Vaswani et al. "Attention Is All You Need." In: *arXiv:1706.03762 [cs]* (Dec. 2017). arXiv: 1706.03762 [cs].

[196] Ashish Vaswani et al. "Attention Is All You Need." In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[197] Ting Hei Wan et al. "Anomaly Detection of Train Wheels Utilizing Short-Time Fourier Transform and Unsupervised Learning Algorithms." In: *Engineering Applications of Artificial Intelligence* 122 (June 2023), p. 106037. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2023.106037.

[198] Biao Wang et al. "A Hybrid Prognostics Approach for Estimating Remaining Useful Life of Rolling Element Bearings." In: *IEEE Transactions on Reliability* 69.1 (Mar. 2020), pp. 401–412. ISSN: 0018-9529, 1558-1721. DOI: 10.1109/TR.2018.2882682.

[199] Biao Wang et al. "Recurrent Convolutional Neural Network: A New Framework for Remaining Useful Life Prediction of Machinery." In: *Neurocomputing* 379 (Feb. 2020), pp. 117–129. ISSN: 09252312. DOI: 10.1016/j.neucom.2019.10.064.

[200] Fu-Kwun Wang and Tadele Mamo. "A Hybrid Model Based on Support Vector Regression and Differential Evolution for Remaining Useful Lifetime Prediction of Lithium-Ion Batteries." In: *Journal of Power Sources* 401 (Oct. 2018), pp. 49–54. ISSN: 03787753. DOI: 10.1016/j.jpowsour.2018.08.073.

[201] Jiang Wang et al. "Learning Fine-Grained Image Similarity with Deep Ranking." In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, OH, USA: IEEE, June 2014, pp. 1386–1393. ISBN: 978-1-4799-5118-5. DOI: 10.1109/CVPR.2014.180.

[202] Qiyao Wang et al. "Health Indicator Forecasting for Improving Remaining Useful Life Estimation." In: *2020 IEEE International Conference on Prognostics and Health Management (ICPHM)*. June 2020, pp. 1–8. DOI: 10.1109/ICPHM49022.2020.9187047.

[203] Yong Wang. "Gauss–Newton Method." In: *WIREs Computational Statistics* 4.4 (2012), pp. 415–420. ISSN: 1939-0068. DOI: 10.1002/wics.1202.

[204] Zezhou Wang et al. "Methods for Predicting the Remaining Useful Life of Equipment in Consideration of the Random Failure Threshold." In: *Journal of Systems Engineering and Electronics* 31.2 (Apr. 2020), pp. 415–431. ISSN: 1004-4132. DOI: 10.23919/JSEE.2020.000018.

[205] Frederik Warburg et al. "Bayesian Triplet Loss: Uncertainty Quantification in Image Retrieval." In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, QC, Canada: IEEE, Oct. 2021, pp. 12138–12148. ISBN: 978-1-66542-812-5. DOI: 10.1109/ICCV48922.2021.01194.

[206] T. Warren Liao. "Clustering of Time Series Data—a Survey." In: *Pattern Recognition* 38.11 (Nov. 2005), pp. 1857–1874. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2005.01.025.

[207] Greg Welch. *An Introduction to the Kalman Filter*. 1997.

[208] Pengfei Wen et al. "A Generalized Remaining Useful Life Prediction Method for Complex Systems Based on Composite Health Indicator." In: *Reliability Engineering & System Safety* 205 (Jan. 2021), p. 107241. ISSN: 0951-8320. DOI: 10.1016/j.ress.2020.107241.

[209] Qingsong Wen et al. *Transformers in Time Series: A Survey*. May 2023. DOI: 10.48550/arXiv.2202.07125. arXiv: 2202.07125 [cs, eess, stat].

[210] Lifeng Wu, Xiaohui Fu, and Yong Guan. "Review of the Remaining Useful Life Prognostics of Vehicle Lithium-Ion Batteries Using Data-Driven Methodologies." In: *Applied Sciences* 6.6 (June 2016), p. 166. ISSN: 2076-3417. DOI: 10.3390/app6060166.

[211] M. Xia et al. "Remaining Useful Life Prediction of Rotating Machinery Using Hierarchical Deep Neural Network." In: *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Oct. 2017, pp. 2778–2783. DOI: 10.1109/SMC.2017.8123047.

[212] Xiaohui Xie. "Principal Component Analysis." In: ().

[213] Dong Xu et al. "Residual Fatigue Life Prediction of Ball Bearings Based on Paris Law and RMS." In: 25.2 (2012), pp. 320–327. ISSN: 10009345. DOI: 10.3901/CJME.2012.02.320.

[214] Mingjing Xu et al. "Fault Prognostics by an Ensemble of Echo State Networks in Presence of Event Based Measurements." In: *Engineering Applications of Artificial Intelligence* 87 (Jan. 2020), p. 103346. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2019.103346.

[215] Yuanjie Yan et al. "Image Clustering via Deep Embedded Dimensionality Reduction and Probability-Based Triplet Loss." In: *IEEE Transactions on Image Processing* 29 (2020), pp. 5652–5661. ISSN: 1057-7149, 1941-0042. DOI: 10.1109/TIP.2020.2984360.

[216] Zijian Ye et al. "Rolling Bearing Health Indicator Extraction and RUL Prediction Based on Multi-Scale Convolutional Autoencoder." In: *Applied Sciences* 12.11 (Jan. 2022), p. 5747. ISSN: 2076-3417. DOI: 10.3390/app12115747.

[217] Cancan Yi et al. "Time-Varying Fault Feature Extraction of Rolling Bearing via Time–Frequency Sparsity." In: *Measurement Science and Technology* 32.2 (Dec. 2020), p. 025116. ISSN: 0957-0233. DOI: 10.1088/1361-6501/abb50f.

[218] Youngji Yoo and Jun-Geol Baek. "A Novel Image Feature for the Remaining Useful Lifetime Prediction of Bearings Based on Continuous Wavelet Transform and Convolutional Neural Network." In: *Applied Sciences* 8.7 (July 2018), p. 1102. DOI: 10.3390/app8071102.

[219] Jianbo Yu. "Local and Nonlocal Preserving Projection for Bearing Defect Classification and Performance Assessment." In: *IEEE Transactions on Industrial Electronics* 59.5 (May 2012), pp. 2363–2376. ISSN: 1557-9948. DOI: 10.1109/TIE.2011.2167893.

[220] Wennian Yu, II Yong Kim, and Chris Mechefske. "An Improved Similarity-Based Prognostic Algorithm for RUL Estimation Using an RNN Autoencoder Scheme." In: *Reliability Engineering & System Safety* 199 (July 2020), p. 106926. ISSN: 09518320. DOI: 10.1016/j.ress.2020.106926.

[221] Alireza Zendehboudi, M. A. Baseer, and R. Saidur. "Application of Support Vector Machine Models for Forecasting Solar and Wind Energy Resources: A Review." In: *Journal of Cleaner Production* 199 (Oct. 2018), pp. 272–285. ISSN: 0959-6526. DOI: 10.1016/j.jclepro.2018.07.164.

[222] George Zerveas et al. "A Transformer-based Framework for Multivariate Time Series Representation Learning." In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. KDD '21. New York, NY, USA: Association for Computing Machinery, Aug. 2021, pp. 2114–2124. ISBN: 978-1-4503-8332-5. DOI: 10.1145/3447548.3467401.

[223] Bin Zhang, Congying Deng, and Yi Zhang. "A Hybrid Feature Selection and Health Indicator Construction Scheme for Delay-Time-Based Degradation Modelling of Rolling Element Bearings." In: *IOP Conference Series: Materials Science and Engineering* 339 (Mar. 2018), p. 012026. ISSN: 1757-8981, 1757-899X. DOI: 10.1088/1757-899X/339/1/012026.

[224] Chi Zhang et al. "Equipment Health Indicator Learning Using Deep Reinforcement Learning." In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Ulf Brefeld et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 488–504. ISBN: 978-3-030-10997-4. DOI: 10.1007/978-3-030-10997-4_30.

[225] H. Zhang et al. "Nonlinear-Drifted Fractional Brownian Motion With Multiple Hidden State Variables for Remaining Useful Life Prediction of Lithium-Ion Batteries." In: *IEEE Transactions on Reliability* 69.2 (June 2020), pp. 768–780. ISSN: 1558-1721. DOI: 10.1109/TR.2019.2896230.

[226] Jiusi Zhang et al. "Prediction of Remaining Useful Life Based on Bidirectional Gated Recurrent Unit with Temporal Self-Attention Mechanism." In: *Reliability Engineering & System Safety* 221 (May 2022), p. 108297. ISSN: 0951-8320. DOI: 10.1016/j.ress.2021.108297.

[227] Qi Zhang and Linfeng Deng. "An Intelligent Fault Diagnosis Method of Rolling Bearings Based on Short-Time Fourier Transform and Convolutional Neural Network." In: *Journal of Failure Analysis and Prevention* 23.2 (Apr. 2023), pp. 795–811. ISSN: 1864-1245. DOI: 10.1007/s11668-023-01616-9.

[228] Zhihua Zhang. *Multivariate Time Series Analysis in Climate and Environmental Research*. Cham: Springer International Publishing, 2018. ISBN: 978-3-319-67339-4 978-3-319-67340-0. DOI: 10.1007/978-3-319-67340-0.

[229] Xiaohang Zhao, Ke Zhang, and Yi Chai. "A Multivariate Time Series Classification Based Multiple Fault Diagnosis Method for Hydraulic Systems." In: *2019 Chinese Control Conference (CCC)*. July 2019, pp. 6819–6824. DOI: 10.23919/ChiCC.2019.8866359.

[230] Lilei Zheng et al. "Siamese Multi-Layer Perceptrons for Dimensionality Reduction and Face Identification." In: *Multimedia Tools and Applications* 75.9 (May 2016), pp. 5055–5073. ISSN: 1380-7501, 1573-7721. DOI: 10.1007/s11042-015-2847-3.

[231] Yuhuang Zheng. "Predicting Remaining Useful Life Based on Hilbert–Huang Entropy with Degradation Model." In: *Journal of Electrical and Computer Engineering* 2019 (Feb. 2019), e3203959. ISSN: 2090-0147. DOI: 10.1155/2019/3203959.

[232] Guo Zhong and Chi-Man Pun. "Subspace Clustering by Simultaneously Feature Selection and Similarity Learning." In: *Knowledge-Based Systems* 193 (Apr. 2020), p. 105512. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2020.105512.

[233] Jianghong Zhou et al. "Remaining Useful Life Prediction by Distribution Contact Ratio Health Indicator and Consolidated Memory GRU." In: *IEEE Transactions on Industrial Informatics* 19.7 (July 2023), pp. 8472–8483. ISSN: 1941-0050. DOI: 10.1109/TII.2022.3218665.

[234] Shuang Zhou et al. "Remaining Useful Life Prediction and Fault Diagnosis of Rolling Bearings Based on Short-Time Fourier Transform and Convolutional Neural Network." In: *Shock and Vibration* 2020 (Oct. 2020), e8857307. ISSN: 1070-9622. DOI: 10.1155/2020/8857307.

[235] J. Zhu, N. Chen, and W. Peng. "Estimation of Bearing Remaining Useful Life Based on Multiscale Convolutional Neural Network." In: *IEEE Transactions on Industrial Electronics* 66.4 (Apr. 2019), pp. 3208–3216. ISSN: 1557-9948. DOI: 10.1109/TIE.2018.2844856.

[236] Enrico Zio. "Prognostics and Health Management (PHM): Where Are We and Where Do We (Need to) Go in Theory and Practice." In: *Reliability Engineering & System Safety* 218 (Feb. 2022), p. 108119. ISSN: 0951-8320. DOI: 10.1016/j.ress.2021.108119.