Constructing explainable health indicators for aircraft engines by developing an interpretable neural network with discretized weights

Moradi, Morteza; Komninos, Panagiotis; Zarouchas, Dimitrios

DOI
10.1007/s10489-024-05981-2

Publication date
2025

Document Version
Final published version

Published in
Applied Intelligence

Applied Intelligence (2025) 55:143
https://doi.org/10.1007/s10489-024-05981-2

Constructing explainable health indicators for aircraft engines by developing an interpretable neural network with discretized weights

Morteza Moradi[1] · **Panagiotis Komninos**[1] · **Dimitrios Zarouchas**[1]

Accepted: 1 October 2024

Abstract
Remaining useful life predictions depend on the quality of health indicators (HIs) generated from condition monitoring sensors, evaluated by predefined prognostic metrics such as monotonicity, prognosability, and trendability. Constructing these HIs requires effective models capable of automatically selecting and fusing features from pertinent measurements, given the inherent noise in sensory data. While deep learning approaches have the potential to automatically extract features without the need for significant specialist knowledge, these features lack a clear (physical) interpretation. Furthermore, the evaluation metrics for HIs are nondifferentiable, limiting the application of supervised networks. This research aims to develop an intrinsically interpretable ANN, targeting qualified HIs with significantly lower complexity. A semi-supervised paradigm is employed, simulating labels inspired by the physics of progressive damage. This approach implicitly incorporates nondifferentiable criteria into the learning process. The architecture comprises additive and newly modified multiplicative layers that combine features to better represent the system's characteristics. The developed multiplicative neurons are not restricted to pairwise actions, and they can also handle both division and multiplication. To extract a compact HI equation, making the model mathematically interpretable, the number of parameters is further reduced by discretizing the weights via a ternary set. This weight discretization simplifies the extracted equation while gently controlling the number of weights that should be overlooked. The developed methodology is specifically tailored to construct interpretable HIs for commercial turbofan engines, showcasing that the generated HIs are of high quality and interpretable.

Keywords Interpretable health indicator · Prognostics and health management · Artificial neural network · Feature fusion · Multiplicative neuron · Ternary weights

1 Introduction

The health indicator (HI) is a crucial index of an engineering system that indicates the status of a component's health in order to make maintenance decisions [1]. A HI can be created based on the necessary information extracted from the sensory data gathered by condition monitoring (CM) techniques. Nevertheless, given the unprocessed nature of all the early data produced by CM techniques, it is likely that they are inappropriate for analysis. Developing HIs from raw sensor data, which are typically uninformative, for diagnostic and prognostic needs is a challenging but essential task [2]. Although a HI is employed by a prognostic model to forecast the remaining useful life (RUL), it brings further value, such as interpretability and a closer connection to the component's health (damage) status. It is important to emphasize that more qualified HI leads to more precise RUL forecasts, which enhances decision-making processes [3].

HIs should satisfy the HI's evaluation criteria in order to be effectively used for diagnostic and prognostic applications. Three primary and well-known prognostic criteria—monotonicity (Mo), trendability (Tr), and prognosability (Pr)—are used to assess the HI's quality [4–8]. The HI's overall increasing or decreasing evolution is indicated by Mo, whereas the distribution of HI values at the end of life (EoL) is quantified by Pr. Tr examines whether the deterioration trajectories of a given component follow a similar underlying trend. Given high criteria scores, even a basic

✉ Morteza Moradi
M.Moradi-1@tudelft.nl

[1] Center of Excellence in Artificial Intelligence for structures, prognostics & health management, Aerospace Engineering Faculty, Delft University of Technology, Delft, The Netherlands

Published online: 13 December 2024

prognostic model can adequately forecast RUL. Tr and Pr calculations need data for **two or more failed components**, meaning more than two HIs from a healthy condition until EoL. This limitation, coupled with the nondifferentiability of the objective function involving Mo, Pr, and Tr, makes it very challenging to directly implement them into a supervised neural network based on backpropagation.

There are two types of HIs: physical (pHIs) and virtual (vHIs) [9, 10]. The first ones are created directly from physical measurements, such as static or dynamic strains, acoustic emission, temperature, displacement, pressure, or a combination of these. The latter are typically designed to provide desired characteristics, including Mo, Tr, and Pr, which greatly increase prognostic efficiency [9], [10], as the first ones are rather unlikely to be sufficient, especially for complicated components. Nevertheless, one important aspect that cannot be addressed using prognostic criteria is the interpretability of a HI. In recent years, a number of data-driven models have been developed to provide a reasonable option for HI [11], [12]. However, the HI functions produced by data-driven models are so complicated that they are almost beyond comprehension, i.e., they lose their physical meanings [13]. Sometimes it is even challenging to translate the raw data from a sensor into a physical phenomenon, let alone interpret the output of a complex data-driven model run on a network of various sensors. Thus, the first step towards understanding the physics of a system could be made if the interpretability of the fusion model of the sensory inputs—i.e., understanding which sensors were used and how they formed the output (HI)—is possible. Additionally, the overfitting of a function decreases with increasing interpretability. With this in mind, the main contribution of this research is the development of a model to cope with the challenge of designing an interpretable HI.

In applications with massive amounts of data, artificial neural networks (ANNs) and deep learning (DL) algorithms can be utilized to automatically build HIs without requiring a lot of domain knowledge. On the other hand, due to the thousands or even millions of parameters required by an applicable ANN for generating HIs, the features produced by DL are complex to interpret and subsequently cannot be treated as physical characteristics of the system/component under monitoring. In fact, data-driven approaches (such as ANN, reinforcement learning, etc.) offer little insight into the relationship between the inputs of captured sensory data and the outputs (either HI or RUL) of the model (black-box) [14]. One of the main causes is that DL models typically have thousands of parameters [15], e.g., when generating a HI, which makes them less generalized and very complicated (the formula behind the DL model is not readable). This renders them inefficient in terms of interpretability (i.e., the equations are overly lengthy, involving numerous variables and parameters).

However, it is important to define the term interpretability first. In light of this, we allocate the next section to explore interpretability, particularly in the context of prognostics and HI application, where the two primary aspects of machine learning interpretability—post—hoc and intrinsic—are discussed. At the end of the next section, we present our methodology, including the key contributions.

2 Interpretability in machine learning for prognostics

2.1 Machine learning interpretability

The field of machine learning (ML) often grapples with the challenge of "interpretability," which lacks a universally accepted definition. Instead, multiple dimensions of interpretability are emerging, each quantifiable in different ways [16]. Interpretable ML (iML) and eXplainable AI (XAI) have grown significantly, focusing on making AI systems more transparent and understandable [16, 17]. Explainability and interpretability are different, even if they are related. iML concentrates on the model itself, whereas XAI concentrates on the model's output [18]. Additionally, iML has been referred to as intrinsic interpretability to distinguish it from methods of post-hoc explainability [16, 19]. From an analytical perspective, post-hoc techniques are the main focus of XAI [20], as they shed light on potentially unreadable black-box models. The term "white-" or "gray-box" ML models is described by iML [21], and architectural or functional restrictions impose restrictions on how they can be interpreted. iML is analogous to models that do not require an explainable mechanism to translate them; instead, they can be understood as being intrinsically interpretable through techniques like physical connections, structural constraints, etc.

Recent years have seen an increase in interest in the application of ML interpretability in the context of prognostics and health management (PHM), predictive maintenance, and digital twins [20, 22]. As reported in a comprehensive review [22], XAI gained more attention than iML in PHM, where prognostic topics gained less attention than diagnostic ones. Among prognostic topics, HI construction is the least explored, while there are a few works on RUL predictions, especially using XAI. Thus, the need for such studies is evident, as HIs carry valuable knowledge of the system under monitoring and play an important role in the field of PHM.

In the following, after briefly explaining ML interpretability in two main categories, we narrow our approach. As may already be noticed, approaches to ML interpretability are broadly categorized into two types: intrinsic and post-hoc. Intrinsic interpretability is achieved by designing models that are inherently understandable, whereas post-hoc

interpretability involves analyzing and explaining models after they have been trained [16].

2.2 Post-hoc interpretability

Post-hoc interpretability seeks to comprehend complicated models after they have been trained. This strategy has gained interest in the context of RUL prediction, where techniques like SHAP (SHapley Additive exPlanations) [23–25], LIME (Local Interpretable Model-agnostic Explanations) [24, 26, 27], and Grad-CAM (Gradient-weighted Class Activation Mapping) [28] are employed to offer insights into model behavior and feature importance. These approaches, however, have several drawbacks:

- Approximation: Post-hoc explanations frequently make assumptions about the model's decision-making procedure, which could result in erroneous interpretations [29].
- Local interpretations: While many post-hoc techniques offer local explanations, they may not generalize well across various input data points.
- Complexity: These explanations can be challenging to follow, particularly when dealing with complicated signals such as multivariate time series collected by sensors.
- Sensitivity: Post-hoc approaches' susceptibility to even minor perturbations in input data raises questions regarding their reliability [30, 31].

Similar or even more severe concerns can arise in HI construction, where the non-transparency and complexity of post-hoc explanations impede their practical applicability in safety-critical contexts such as aircraft engine health monitoring. Since there are no actual labels for HI, which limits supervised ML models, there may be more obstacles in HI design than in RUL prognostics [32].

2.3 Intrinsic interpretability

To achieve intrinsic interpretability, models must be created that are naturally comprehensible. This method ensures that the model's decision-making process is transparent and easily comprehensible for humans. Simple decision trees and sparse linear models, for example, are frequently regarded as intrinsically interpretable because their whole structure can be understood at once. However, even these models can become opaque if they are very high-dimensional or complicated [29]. While choosing a simple hypothesis class, such as linear models or small decision trees, is a straightforward way to achieve interpretability, this strategy frequently comes at the expense of predictive performance, which is not always acceptable [33].

In this regard, a model should not only result in a prediction close to the target value but also should consist of a few features [34]. This directly relates to interpretability because humans can only comprehend processes with a finite set of variables [35]. Lipton [29] argues that neither linear models, rule-based systems, nor decision trees are inherently interpretable in all cases. Reasonably compact neural networks can become more transparent than sufficiently high-dimensional models, unmanageable rule lists, and deep decision trees. Therefore, to preserve transparency while balancing predictive performance, establishing intrinsic interpretability necessitates careful consideration of model complexity and structure.

Among intrinsically interpretable ML techniques, symbolic regression, a branch of regression analysis, aims to discover symbolic expressions that accurately fit a given dataset [36]. This method yields interpretable equations that characterize the data by inferring both the model structure and parameters in a data-driven manner. As the length of the string (arithmetic, symbolic expressions), in addition to the presence of numeric constants, increases, the function space of symbolic regression exponentially increases. This complicates the exploration process, making symbolic regression especially challenging [37]. One way to overcome this is to model it as an optimization problem with a gradient basis.

Recent advancements have employed genetic programming (GP), the most popular symbolic regression approach, enabling the discovery of physical laws and transparent RUL regression models [38, 39]. Recently, GP has been successfully adopted into the CMAPSS dataset in a model called two-stage GP for aircraft engine health monitoring, demonstrating its potential for generating interpretable HIs [40].

2.4 Our approach

Demonstrating that a new model accurately reflects the original model is the final goal of many fields, which is indeed challenging, particularly for complex systems like turbofan engines. Here are some steps to achieve this (with a focus on HI), where interpretability plays an important role:

- Extracting a readable equation: The first step is to extract (discover) a readable equation that describes the relationship between inputs and outputs of the model, which should align with the physical principles governing the system under study. It is essential to see how the inputs influence the model's output, including the direction (direct or inverse relationship), the severity (e.g., the power of the input's effect on the output), and the function of severity (e.g., whether the effect is linear, polynomial, exponential, etc.), which all are met in a readable equation. In XAI, the direction can be mainly addressed, while

iML targets all three as it seeks a model that is intrinsically interpretable.

- Confirming physical phenomena: As mentioned in the first step, the extracted equation should align with the physical principles governing the system under study, which it deems to reflect the original model. For HI, it is first crucial to meet prognostic criteria such as Mo, Pr, and Tr, among which Mo and Pr are factual measures, while Tr is ideal and depends on how and when uncertain phenomena happen to the unit under monitoring. The next level of validation in this step is that a HI should also meet diagnostic requirements, meaning it should accurately reflect the physical events affecting the unit during monitoring and correlate with the severity of those events in a synchronized way. Diagnostic requirements are not independent from prognostic ones. The accumulation of the severity of those events is measured by Pr, demonstrating the same final failure. The synchronization of those events with what the equation shows is somehow related to Tr. On the other hand, to observe and validate the latter, different measurements and labels for each unit (turbofan engines in this study) are required over operational time, which is expensive and sometimes not feasible. To sum up, the extracted equation and its variables should be explained in the context of physical phenomena, ensuring that the model's behavior is consistent with known physical laws and observations.

Given the limitations of post-hoc interpretability and the advantages of intrinsic interpretablity, our research focuses on developing an intrinsically interpretable ML technique for HI construction. We propose a novel method for constructing interpretable HIs for aircraft engines by developing an interpretable neural network with discretized weights. This model aims to provide clear and straightforward mathematical functions that relate sensory data to HIs, ensuring transparency and ease of understanding, i.e., in the strictest sense (transparency), a person can contemplate the entire model at once.

Typical ANNs employ additive neurons, which multiply inputs by weights before summing the outputs. As a consequence, the option to multiply the inputs together is missing, particularly in situations where numerous inputs are involved, such as CM sensory data. Instead of only taking additive neurons into account, the multiplicative operator may produce a simpler, more inclusive, and more understandable equation. In fact, a multiplicative operator can supply multiple summation operators, resulting in a shorter length for the output equation. For instance, the HI function developed by [40], for the CMAPSS dataset, uses just the multiplication and division operators between the features—there is no use of the summing operator. In order to simulate multiplication and division operations using purely summing operators (assuming this is doable), it is most likely necessary to use more weighted summation operators, which would make the HI function more complicated to comprehend.

In this paper, a multiplicative neuron is first developed to be exploited alongside the typical additive ones. The suggested multiplicative neuron is adaptable since it can multiply all or part of the inputs together and is not restricted to multiplying them in pairs [41]. Division operations can also be carried out simultaneously, albeit it should be mentioned that using division units in a neural network is more challenging [42]. Then, a HI will be designed by integrating additive and multiplicative neurons. However, this modification alone is not enough to extract a compact HI from the ANN. Having continuous weights for each neuron still makes the equation large and non-understandable, with possibly many unnecessary terms that ought to be removed. In this respect, the weights are discretized in two ways: first, using a ternary set, and second, softening the ternary set by rounding the weights at the first decimal number. This is performed in order to develop a simple yet effective formula for the HI and maintain its interpretability. The ternary set is constructed in a fashion to converge the weights towards the values `{-1, 0, 1}` by controlling the number of zeroes as well, where the latter is termed **sparsity control** [43]. In other words, an ANN consisting of many additive and multiplicative neurons with discrete weights is proposed, where most of them can be controlled by converging towards zero (sparsity control) during training. As such, having both the summation and multiplication operators as well as the discretization with sparsity control makes the equation simple and the ANN interpretable.

Given that HIs' criteria are non-differentiable and the entire timeseries of HIs, and consequently, CM data up to the EoL should be available to calculate them— which is impractical in many applications—a variety of kernel functions (linear, polynomial, logarithmic, and exponential) were explored based on their suitability to simulate hypothetical HIs [44]. The investigation revealed that the polynomial kernel equation provided the highest prognostic criteria, inspired by damage accumulation [45]. Consequently, a label simulation technique under the inductive semi-supervised learning (SSL) paradigm can be adopted to train a back-propagation-based model.

To validate our approach, we compare our model with several established methods, including principal component analysis (PCA), kernel PCA (KPCA), and the two-stage GP model. The first two techniques are considered for comparison as they are capable of generating qualified HI in many applications. Nevertheless, as already mentioned, they lack interpretability since, for high-dimensional inputs, the length of the output equation is long and unreadable, i.e., it is not sparse. Moreover, in the case of KPCA, the yielded equation is not only sparse (it is longer than that of PCA) but also more complicated as it is involved in kernel functions. The

two-stage GP model was recently applied to the CMAPSS dataset to generate interpretable HIs for commercial turbofan engines while maintaining prognostic quality [19].

The proposed methodology is specifically designed to construct interpretable HIs for commercial turbofan engines, utilizing the NASA Ames Prognostics Data Repository dataset, a widely recognized resource in the PHM field [46]. The findings will be discussed in comparison with PCA, KPCA, and two-stage genetic programming (GP) outputs. This work makes several key contributions, including:

1. Introducing a new type of neuron that operates multiplicatively and is not limited to pairwise operation, in addition to the commonly used additive neurons.
2. Building a network that combines both additive and multiplicative neurons to generate accurate and robust hybrid HI.
3. Utilizing the benefits of both multiplicative neurons and sparsity control by implementing discretized (ternary) weights, resulting in concise and efficient equations.
4. Developing HI models with concise and easy-to-understand equations, while ensuring they meet the evaluation criteria of monotonicity, trendability, and prognosability.
5. Illustrating which sensors are involved in the HI of turbofan engines and how effective these sensors are (considering their direct (positive) or inverse (negative) relationship and severity that can be measured by the power in the equations).
6. Estimating the threshold for the time-to-failure of turbofan engines with a minimum accuracy of 97.8% considering the end of life (EoL) and, 93.2% for higher safety confidence, i.e., 5% earlier than the end of life.

In summary, our work addresses the need for HI construction in aircraft engines by leveraging the strengths of intrinsically interpretable ML models. By comparing our model with both GP and conventional methods like PCA and KPCA, we aim to demonstrate its superiority in providing clear and interpretable health indicators.

The remainder of the paper is divided into three sections, including Section 3 Workflow, Section 4 Results and discussions, and Section 5 Conclusion.

3 Workflow

First, a short description of the pre-processing, de-noising, and data division into training and test are presented in the Dataset subsection (3.1). Next, the HI construction method (3.2) containing the additive neuron, the multiplicative neuron, discretized weights, and building the interpretable ANN (INN), is described. Finally, the health indicator's evaluation criteria (3.3) will be provided.

3.1 Dataset

The present study focuses on the NASA Ames Prognostics Data Repository dataset for commercial turbofan engines (CMAPSS) [46]. This dataset is generated using the C-MAPSS tool, which models various engine fleet deterioration scenarios—from a baseline condition to the point of final failure in the training data and a time period prior to the EoL in the test data. The research investigates two sets of data: first, engines degrading with one failure mode (FD001); and second, engines degrading with two failure modes (FD003). Each engine's ID and deterioration time steps are given in the first and second columns, respectively. The next three columns provide the engine's operational characteristics, and the final 21 columns list the signals from 21 sensors. Both subsets FD001 and FD003 consist of 200 turbofan engine units each, with 100 designated for model training and the remaining 100 for testing RUL prediction models. However, the 100 units allocated for RUL model testing lack sensory data up to the EoL. Consequently, these units cannot be utilized for evaluating the HI construction model using the prognostic criteria (Mo, Tr, and Pr). As a result, a test fraction equivalent to 20% of the first 100 turbofan engines with complete input sensory data until EoL in each subset is reserved for testing.

Data processing can suffer from signals that are constant throughout all measurement points. As a result, data that have identical upper and lower boundaries is first found and removed. Accordingly, out of the 21 sensors, the 1^{st}, 5^{th}, 10^{th}, 16^{th}, 18^{th}, and 19^{th} are removed, leaving 15 in place for the subset FD001. The same sensors are removed from subset FD003 with the exception of sensor 10, leaving 16 in place. As a result, the remaining sensors are denoted in the following as 1 through 16, among which the 16^{th} sensor refers to the different sensor used in subset FD003 (i.e., sensor 10), and the other sensors have the same index. Data have been standardized using a zero-mean normalization technique that used only the training samples' mean value and standard deviation. Additionally, to improve the quality of the resulting features and HI, the signals can be de-noised. In this case, a polynomial function of order four is used to perform a regression. The resulting de-noised signals (features) can then be chosen as HIs or retrieved (feature extraction) and combined (feature fusion) to create an appropriate HI.

3.2 HI construction method

Recent research has demonstrated that it is feasible to build HIs simply by adding and multiplying the derived features from sensory data [40]. In this section, we present the idea

of creating such mathematical operators inside the ANN automatically in order to formulate simple yet efficient HIs without compromising the high accuracy that deep learning may provide. It should be highlighted that the equation is not produced as an output by the ANN; rather, the ANN contains the equation itself.

In the context of learning paradigms, the model is tasked with optimizing an objective function that includes prognostic criteria (Mo, Pr, and Tr). However, this function is non-differentiable, posing challenges for implementation in backpropagation. Consequently, a semi-supervised learning framework is employed, implicitly incorporating HI criteria due to the absence of true labels for HIs [47]. In this methodology, prognostic metrics are utilized to identify the optimal simulator for generating ideal labels (targets), which are subsequently used to guide the model. The ideal simulator function ($HI_{(t)} = t^2/t^2_{EoL}$), which is in terms of the operational time (t), has a quadratic polynomial shape.

A limited number of neurons and layers should be employed in order to derive an effective equation that could characterize a HI. An ANN's compact size depends on the subject under examination. It is presumed that even a basic network of two 8-neuron layers could result in an outsized, physically unexplainable equation representing a HI. At first, it can seem extremely difficult for an ANN to be trained with just a few parameters and deliver correct results. A probable underfitting of the data is foreseen, even for small datasets. A straightforward HI equation can be derived from the ANN itself by including the physical parameters in it and zeroing out some weights in the training step. In the current work, it is regarded that physical properties could be basic multiplications and summations among features, which can be done by the combination of the multiplicative and additive layers, as will be seen in the next subsections. By discretizing the weights into a ternary shape and regulating the number of weights that should be zero, as will be explained in Section 3.2.3, it is technically feasible to automatically decrease the number of neurons and further simplify the HI formula.

3.2.1 Introspecting ANN - additive neuron

Artificial neurons, which are coupled together and organized into layers, are the building blocks of an ANN. Each layer receives input from signals. One layer's output feeds into the subsequent layer's input. The basic equation of the typical ANN for each neuron individually given certain inputs x_K from the preceding layer is:

$$N_j = \sum_{i=1}^{K} [w^l_{ji} x_i] + b^l \tag{1}$$

where N_j is the initial output of neuron and w^l_{ji} is the weight relevant to the connection between the j^{th} neuron at the l^{th} layer to the $(l-1)^{th}$ layer's i^{th} neuron. The neuron also contains b^l to consider the bias. By using a nonlinearity through an activation function $F(N)$, which has the only restriction of being differentiable to the points of interest, the final output of the neuron is computed. While the ANN is being trained, the weights and biases of each neuron, which stand in for the learnable parameters of the network, are attempting to modify their values by backpropagating the error through the derivatives. A differentiable loss function must also be formulated at the points of interest. Due to the fact that this neuron sums the weighted inputs, it is termed **additive**.

3.2.2 Introspecting ANN - multiplicative neuron

The basic equation of the neuron (1) should be modified in order to induce the layers to generate multiplication operators. Accordingly, as stated in [48], we can construct a **multiplicative** neuron instead of a typical additive neuron by changing the summation step ($\sum_{i=1}^{K}[w_{ji}x_i]$) to a multiplication one ($\prod_{i=1}^{K}[x_i^{w_{ji}}]$), with the weights acting as exponents in a product rather than weights in a sum. Unfortunately, as mentioned in [49], an ANN with typical multiplicative layers makes the training more complex and slower due to the derivatives that are needed for backpropagation. This is the main reason that these layers have not been applied extensively in the literature. To mitigate this pitfall, a modified multiplicative neuron is developed by converting the additive neuron via a specific pair of continuous activation functions. In particular, the inputs must get a logarithmic activation before being fed into (1) and an exponential activation afterward. The following equation can be used to update an additive neuron into a multiplicative one after the above-mentioned adjustments:

$$\begin{aligned} N_j &= e^{\sum_{i=1}^{K}[w^l_{ji} ln(x_i)] + b^l} \\ &= e^{b^l} . e^{\sum_{i=1}^{K}[ln(x_i)^{w^l_{ji}}]} \\ &= e^{b^l} . e^{ln(\prod_{i=1}^{K}[x_i^{w^l_{ji}}])} \\ &= e^{b^l} \prod_{i=1}^{K}[x_i^{w^l_{ji}}] \end{aligned} \tag{2}$$

The transition from additive to multiplicative neurons is shown in Fig. 1. An ANN can avoid adding further nonlinearities that might result in a complex equation by only employing these two types of activation functions. A key point to highlight is that by constraining the neurons to perform these particular activation functions, their ability to scale is confined by the requirement that the inputs be positive in order to apply the logarithm. Nonetheless, as the inputs could be simply rescaled to a desirable positive range, this is not a limitation in the current research. Furthermore, the con-

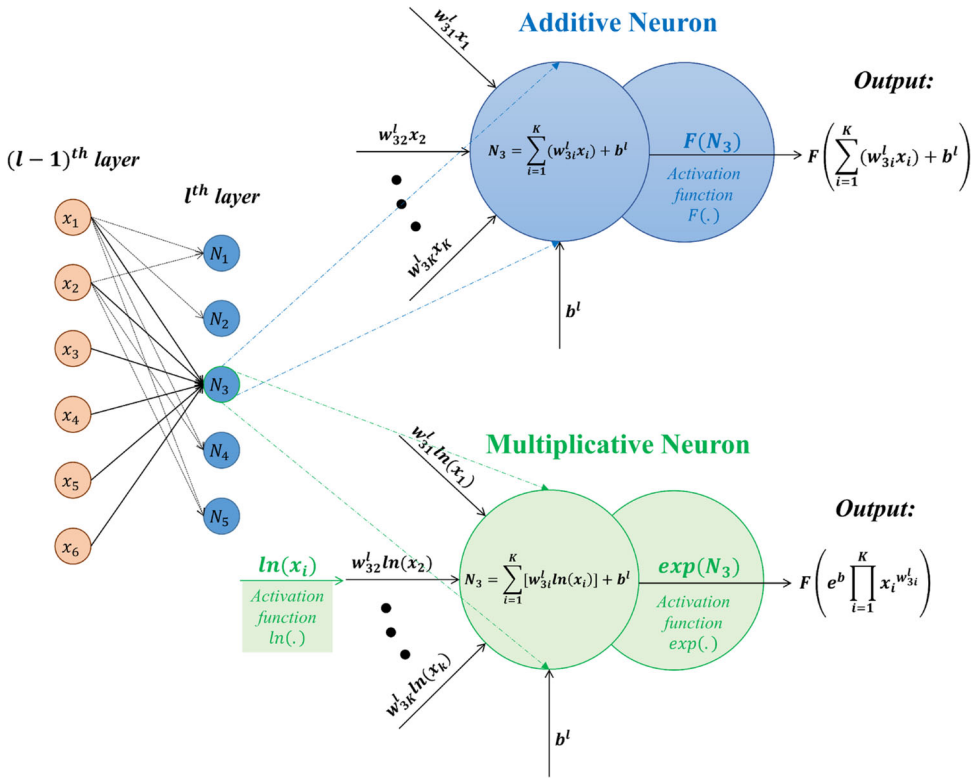


Fig. 1 Additive and multiplicative neurons

vergence principles of neural networks are fulfilled because the logarithm exists and the proposed multiplicative neuron derives naturally from the additive one by adjusting the activation functions.

Ultimately, with this modification, the forward and backward remain the same as the vanilla ANN. During the forward pass, the inputs are fed into the network, and the activations are computed layer by layer until the output is obtained. For a network with L layers, the forward pass can be represented as:

$$\begin{aligned} z^{(1)} &= W^{(1)}x + b^{(1)} \\ a^{(1)} &= \sigma(z^{(1)}) \\ z^{(2)} &= W^{(2)}a^{(1)} + b^{(2)} \\ a^{(2)} &= \sigma(z^{(2)}) \\ &\cdots \\ z^{(L)} &= W^{(L)}a^{(L-1)} + b^{(L)} \\ a^{(L)} &= \hat{y} = \sigma(z^{(L)}) \end{aligned} \tag{3}$$

where x is the input, $W^{(l)}$ and $b^{(l)}$ are the weights and biases of layer l, σ represents the activation function, and $\hat{y}$ is the predicted output. For the multiplicative neuron, the activation function is strictly the exponential one which does not affect the typical derivatives of the backpropagation.

The backward pass involves calculating the gradients of the loss function with respect to the weights and biases of the network using the chain rule of calculus. This process starts from the output layer and propagates the error backward through the network. The gradients of the loss function $\mathcal{L}$ with respect to the weights and biases can be computed using the chain rule as follows:

$$\begin{aligned} \delta^{(L)} &= \frac{\partial L}{\partial \hat{y}} \cdot \sigma'\left(z^{(L)}\right) \\ \frac{\partial L}{\partial W^{(L)}} &= \delta^{(L)} \cdot \left(a^{(L-1)}\right)^T \\ \frac{\partial L}{\partial b^{(L)}} &= \delta^{(L)} \end{aligned} \tag{4}$$

Similarly, for hidden layers $l = L-1, L-2, \ldots, 2$:

$$\begin{aligned} \delta^{(l)} &= \left((W^{(l+1)})^T \delta^{(l+1)}\right) \odot \sigma'(z^{(l)}) \\ \frac{\partial L}{\partial W^{(l)}} &= \delta^{(l)} \cdot \left(a^{(l-1)}\right)^T \\ \frac{\partial L}{\partial b^{(l)}} &= \delta^{(l)} \end{aligned} \tag{5}$$

where $\odot$ represents element-wise multiplication and σ' is the derivative of the activation function. Finally, the weights and biases are updated using an optimization algorithm such as gradient descent:

$$W^{(l)} := W^{(l)} - \alpha \frac{\partial \mathcal{L}}{\partial W^{(l)}} \tag{6}$$

$$b^{(l)} := b^{(l)} - \alpha \frac{\partial \mathcal{L}}{\partial b^{(l)}} \tag{7}$$

where α is the learning rate.

3.2.3 Discretized weights

Learning the weights in continuous spaces is very favorable since gradient-based optimization techniques can be employed to achieve a stable training process that is guaranteed to converge towards an optimal solution. Nevertheless, because the ANN design is typically complex with vast numbers of weights, this is not effective in developing concise equations. Having millions of parameters makes the interpretation hard, and understanding the significance of each weight and its impact on the model's behavior may be difficult. Employing the weights in discrete spaces offers more compact equations, and it is easier to analyze the contribution of individual features to the model's output. This is particularly important in the case of HI construction, where the objective is to recognize the pattern and reconstruct a HI that offers high criteria scores (Mo, Tr, and Pr) rather than merely exact target values [44]. Incorporating continuous values with multiple decimal digits provides a complex model, even in extreme situations where only a small number of weights are non-zero. An ANN cannot be trained with discontinuous weights because there are no gradients for back-propagation; hence, learning in a continuous space is ultimately inevitable. Rounding the weights to the desired decimal during testing could be an easy way to achieve a compact formula, yet it would adversely affect the outputs and most likely result in the ANN being ineffective.

By utilizing ternary weights, we enable straightforward additions and multiplications among input features, resulting in compact equations. For instance, the equation $X_1^{w_1} X_2^{w_2} + w_3 X_3 + w_4 X_4$ can be transformed into $X_1^1 X_2^{-1} + 0X_3 + 1X_4 = X_1/X_2 + X_4$. This demonstrates that ternary weights allow for multiplication, division, or the exclusion of features, thereby enhancing interpretability.

The weights should preferably be discrete to particular decimal values or even integers without compromising precision. Ternary weights have lately been developed to help with this challenge [43]. The objective is to train an ANN by converging the weights to specified values instead of rounding them to particular decimal digits. They are known as "ternary" values if the provided values are `{-1, 0, 1}`. There are undoubtedly scenarios where we require weights to fall within the range of those integers. This approach only induces a portion of the weights, which is controllable, to be integers rather than forcing all.

According to [43], the full-precision weight space is too vast to identify an acceptable ternary solution, thereby the continuous weight spaces need to be constrained by $tanh(w)$:

$$w' = tanh(w) \tag{8}$$

The weights are now bounded to the chosen `[-1, 1]`, hyperbolic tangent range. Adding just one more term to the loss function (E) enables this transition work:

$$E = E_C(y, \hat{y}) + \lambda E_R(w') \tag{9}$$

$$E_C(y, \hat{y}) = \frac{1}{n} \sum_{j=1}^{n} (y_j - \hat{y}_j)^2 \tag{10}$$

$$E_R(w') = E_R(tanh(w)) = \sum_{l=1}^{L} \sum_{i}^{|w^l|} [(\alpha - tanh^2(w_i^l)) tanh^2(w_i^l)] \tag{11}$$

where $E_C(y, \hat{y})$ is the mean-squared loss (MSE) between labels y_j and predicted outputs $\hat{y}_j$ over n data points, and $E_R(w')$ is the discretizing loss for converging the weights towards the ternary values. λ is a regularization parameter, L is the number of layers, $|w^l|$ denotes the total number of weights for the l^{th} layer, and α is the shape controller of $E_R()$. λ and α are supplementary hyperparameters that should be adjusted for training the ANN; the first can be seen as a trade-off between the importance of reducing the MSE and ternarizing more weights effectively, and the latter softly controls the number of weights to become zero. The gradients exist and are proven to be minimal at $tanh(w) = -1$, $tanh(w) = 0$ and $tanh(w) = 1$ when $0 < \alpha < 2$ using the aforementioned modifications and loss functions (the proof in [43]). The $E_R(w')$ for different α values is shown in Fig. 2. The number of zeros in the trained weights, which can be monitored to have more or fewer parameters, is the sparsity control key property of (11). This is especially helpful in situations where larger ANN architectures were obtained, yet we still want to have concise formulations for HIs by zeroing (raising α) more weights. By maintaining the weights as close to their ternary shape as appropriate and regulating the proportion of them that should be identical to zero, the ANN is able to generate precise predictions, which is a benefit of the adjustment to the weights and the incorporation of the term to the total loss function E.

3.2.4 Building interpretable ANN (INN)

By definition, an ANN is a function approximator that integrates input data into the expected output using a complicated equation. Since an ANN needs large numbers of weights to build acceptable HIs, retrieving the equation is

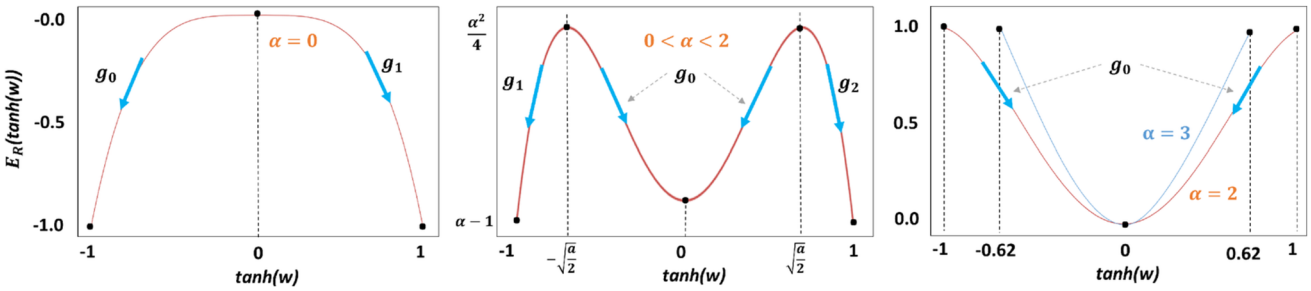


Fig. 2 Gradients' flow of the discretizing loss function E_R during training for different shape controller parameters α. The desired local minima exist when $0 < \alpha < 2$

not practical. The number of variables should be dropped while ensuring high levels of performance to achieve an interpretable network that could be transformed into an intuitive and condensed equation expressing a HI. The proposed methodology will demonstrate that combining the weights' discretization—the regulation of their sparsity—and the simultaneous use of both multiplicative and additive neurons results in an interpretability that is satisfactory. The abovementioned ANN can also take into account the physical characteristics that invisibly underlie the components that make up a HI. As a result, the proposed approach may now uncover the ANN's underlying formula, the HI, which properly represents the feature selection and fusion steps.

An additive layer is composed of several typical additive neurons, whereas a multiplicative layer is made up of several multiplicative neurons. Figure 3 illustrates the proposed framework. First, a multiplicative layer receives the inputs which are either raw sensory data or de-noised ones. Each neuron at this layer is a multiplication of the inputs with various weights and a bias in accordance with (2). There are several multiplication combinations between the inputs when there are plenty of neurons. The output of the multiplicative layer is then added to a subsequent additive layer with a single additive neuron to create the final output. The ANN becomes increasingly complex when more neurons are added to the additive layer, and it is quite probable to overuse a portion of the inputs. Terms that correspond to

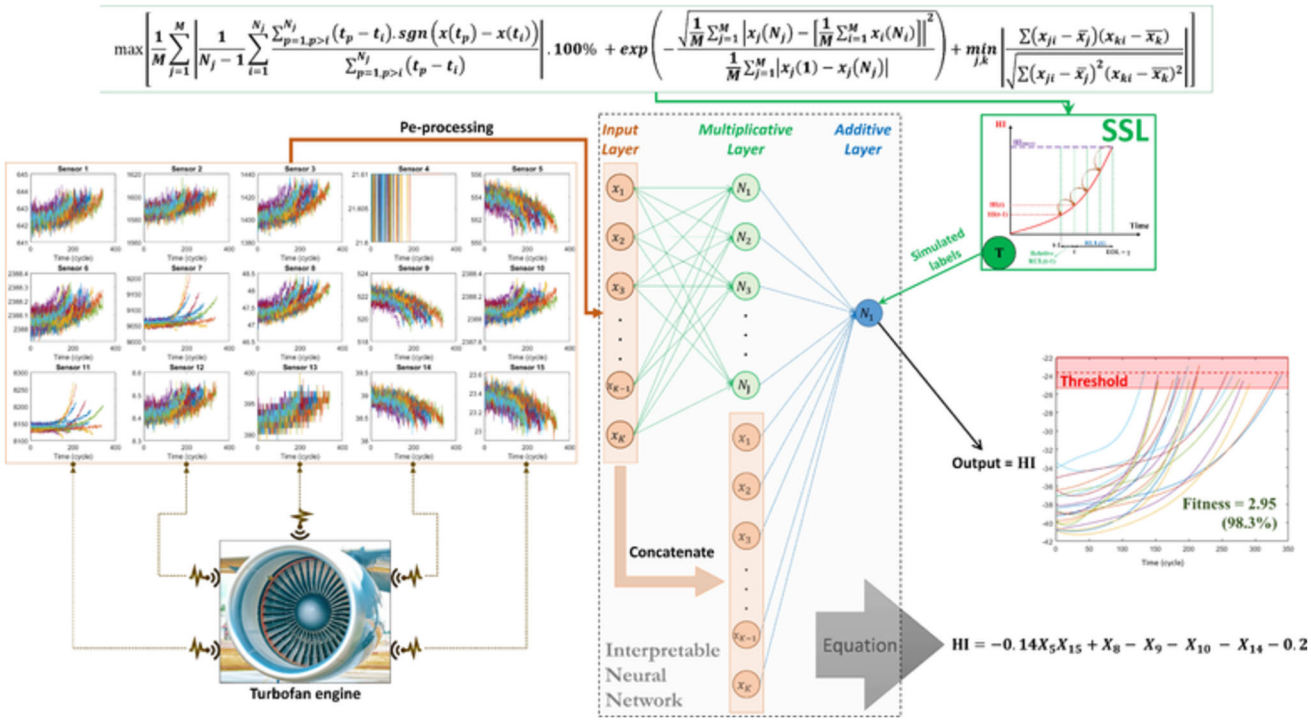


Fig. 3 Framework of the proposed methodology. The inputs are fed into a multiplicative layer, and each neuron applies a multiplication operator. Then, the outputs are concatenated with the inputs and together are driven into the additive layer which consists of one neuron, and the output is alongside the equation of the constructed HI

a single input instead of a combination of them are typically evident once a HI's formula is obtained. For example, if the inputs are x_1, x_2, and x_3, we might get the equation $x_1x_2x_3 + x_1$. It is difficult to establish such an equation utilizing purely the multiplicative layer's outputs to be imported into the subsequent additive layer. Hence, utilizing the inputs of the network in both additive and multiplicative layers is the most applicable architecture. Accordingly, the inputs and outputs from the multiplicative layer are concatenated before being passed into the additive layer. It is clear that using additive layers and then multiplicative layers results in more complex equations, and the option to multiply the inputs together is missing. For instance, considering four inputs (two by two separated (zero weights) for simplification), the simplified combination of additive+multiplicative yields $(x_1 + x_2)(x_3 + x_4) = x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4$ while the simplified combination of multiplicative+additive yields $x_1x_2 + x_3x_4$, which is less complex. This is due to the ability of the multiplication operation to capture more complex dynamics than the addition one. Consequently, the model should first create all the necessary complex combinations between the features and then decide which of them are important to be added together. It should also be noted that, as mentioned, the option to directly multiply the inputs in the additive+multiplicative configuration, i.e., x_1 and x_2 as well as x_3 and x_4, is missing.

A stream of raw sensor data, a de-noised format, or some extracted features could all be used as the model's inputs. The trained INN serves as the equation for generating the outputs, which are a series of points that make the HI. Before importing the data into the model, a preprocessing step including resampling is required due to the different sequence lengths for each unit. The time-series data points will all be the same length thanks to this step. To do this, there are two techniques. The easier method involves upsampling with interpolation, which involves increasing the number of time-series samples until each sequence is equal to the largest one. Based on the interpolation method selected, those data points are estimated. The second technique involves extending each sequence by adding pseudo-data points since it meets the maximum length needed. This can be performed by padding with an irrelevant value. The padded inputs can then be used to train the model, which outputs the results. This method's sensitive aspect is when it calculates losses. To prevent these pseudo-values from biasing errors through backpropagation, the padded lengths should be perfectly deleted. To continue the next forward pass following the updating of the parameters, the lengths should be padded again. This method eliminates the need for any estimation steps, unlike the first technique. Nevertheless, training time rises remarkably. It was found that both techniques in the present work produce outcomes that are equal, and the first one has been adopted thanks to its simplicity and speed.

So far, the equation for formulating the HI was not completely expressive because the weights could have any real value. Thanks to (8) to (11) for training the INN, the majority of the weights, if not all, shift in the direction of the integers -1, 0, or 1. Weights can converge to the intended values in practice, but they may not always coincide. With this in mind, it is safe to appropriately round the values during validation without compromising accuracy in these circumstances. Utilizing a de-noised version of the sensor data enables all the weights to become ternary (see Section 4); however, using their raw version does not cause this to happen. As long as the majority of the weights are within the ternary form, a few weights in this last scenario could range from `[-1, 1]`, which could be smoothly rounded to the first decimal point with a trivial accuracy loss. Following training, several non-ternary weights are a result of the noisy raw data. Hence, there is indeed a trade-off between ternarizing the weights and minimizing the E_C loss, which can be controlled by the regularization hyperparameter λ. A large value for λ indicates that more ternary weights have been preferred (better E_R minimization), leading to a more concise equation rather than optimally predicted results (not ideal E_C minimization). Fortunately, we intend to construct a HI that delivers high criteria scores (Mo, Tr, and Pr), instead of purely simulated label values, thus emphasizing more on developing concise formulas.

3.3 Health indicator's evaluation criteria

To regard a HI as a prognostic indicator, a set of criteria must be satisfied. The three main metrics for assessing a HI are Mo, Pr, and Tr [4], which are formulated as follows:

$$Mo = \frac{1}{M}\sum_{j=1}^{M}\left|\frac{1}{N_j-1}\sum_{i=1}^{N_j}\frac{\sum_{p=1,p>i}^{N_j}\left(t_p-t_i\right).sgn\left(x_j(t_p)-x_j(t_i)\right)}{\sum_{p=1,p>i}^{N_j}\left(t_p-t_i\right)}\right|.100\% \tag{12}$$

$$Tr = \min_{j,k}\left|\rho(x_j, x_k)\right|; \quad j,k = 1,2,...,M \tag{13}$$

$$Pr = exp\left(-\frac{\sqrt{\frac{1}{M}\sum_{j=1}^{M}\left|x_j\left(N_j\right)-\left[\frac{1}{M}\sum_{i=1}^{M}x_i\left(N_i\right)\right]\right|^2}}{\frac{1}{M}\sum_{j=1}^{M}\left|x_j\left(1\right)-x_j\left(N_j\right)\right|}\right) \tag{14}$$

where M is the number of components being monitored, N_j and x_j are the number of observations and the vector of HI on the j^{th} sample. μ, σ, ρ, and sgn are the mean value, standard deviation, Pearson's correlation, and sign functions, respectively. The three HI criteria result in scores in a range of `[0, 1]`, with 0 being the worst and 1 denoting the highest HI quality. t_p and t_i represent the measurement times for $x_{(t_p)}$

and $x_{(t_i)}$, correspondingly. An objective function termed "Fitness", [32, 44] is utilized to simultaneously take into account all of the aforementioned prognostic metrics:

$$Fitness = Mo_{(HI)} + Tr_{(HI)} + Pr_{(HI)} \tag{15}$$

where the fitness index falls between `[0, 3]`, with 0 denoting the lowest HI performance and 3 denoting the highest possible.

The availability of all degradation histories up to EoL is necessary for these criteria to be taken into account. If not, it is impossible to measure Tr and Pr properly. It is important to emphasize that the RUL prediction metrics obtained from prognostic models applied to the HIs in the current study are not perfectly suitable for measuring the success of HI construction models. This is due to the fact that the performance of the prognostic model, not just the HI's quality, has an impact on the RUL prediction metrics.

4 Results and discussions

In this section, the HIs constructed using the introduced approach are compared and discussed with the outputs of the PCA, KPCA, and GP models for subsets FD001 and FD003, respectively.

Thanks to the sparsity control of the ternary weights, the active number of neurons in the INN that contribute to the output can be reduced dramatically. Hence, the number of multiplicative neurons could be large enough to capture multiple combinations of the inputs before extracting a compact HI equation. The weights are uniformly initialized in the range `[-1, 1]` to enable the INN to translate its continuous weights into their ternary version. As will be addressed later, using the raw sensory data requires softening the ternary discretization to a smoother form, where float discrete values with one decimal confined to the same range can be employed.

The INN model was trained on a single GPU (NVIDIA GeForce RTX 2080). The entire training process takes approximately 15 seconds per run. Utilizing the grid search method for hyperparameter tuning increases the total time up to 75 hours. Despite this, the efficiency of our model's training process ensures that we can explore various hyperparameter configurations without excessive computational costs, maintaining robustness, and optimizing performance effectively.

4.1 Subset FD001

Since the first principal component (PC) of the PCA and KPCA covers the largest portion of variations in data, it can be regarded as HI. These PCs were extracted through models applied to the whole dataset including 100 units, which are shown in Fig. 4. Sensor 8 has the highest fitness

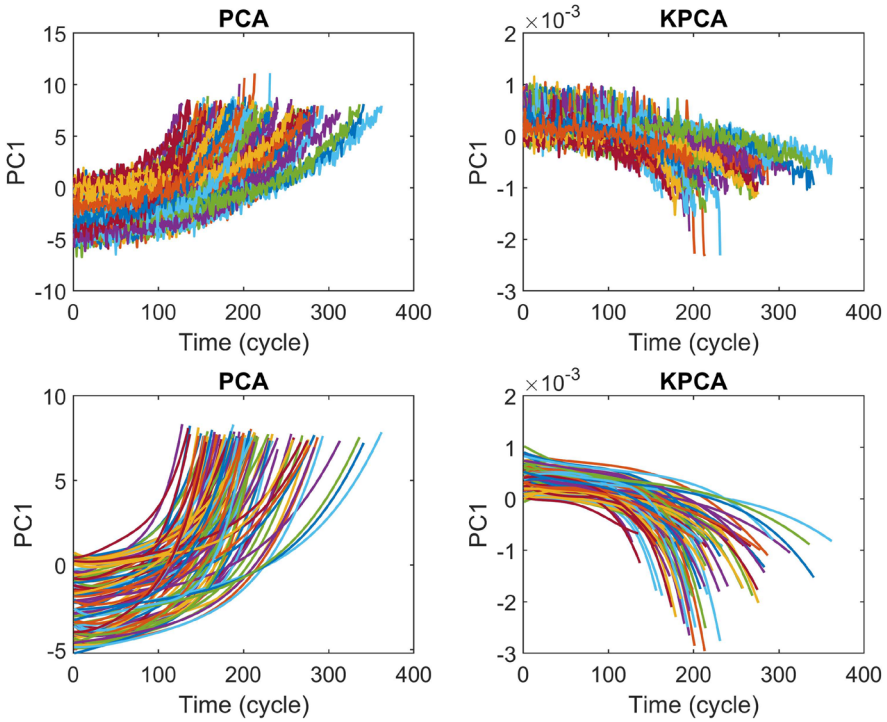


Fig. 4 First principal component of the PCA and KPCA applied to the raw (first row) and de-noised (second row) entire dataset of subset FD001 given 100 engine units

Table 1 The INN model's hyperparameters

Dataset	Alpha (α)	Lambda (λ)	Batches	Epochs	Multiplicative Neurons	Additive Neurons	Learning Rate
de-noised	1.6	10^{-5}	4	200	32	1	0.01
raw	1.8	10^{-3}	4	200	64	1	0.01

score for raw inputs, 2.58, which has been improved using the PCA model to 2.85 (10.47%). This value for de-noised inputs increased from 2.91 (sensor 8) to 2.94 (1%). On the other hand, the KPCA model was unable to improve the HI with regard to neither raw nor de-noised inputs, demonstrating that the CMAPSS dataset (especially, subset FD001) has a linear rather than a nonlinear correlation among inputs. As a result, PCA can produce a reasonably appropriate HI for this dataset, and the results argue that complex models for CMAPSS, including deep neural networks, are unnecessary and redundant. This is also valid for RUL prediction because a better HI results in a more precise RUL forecast. This viewpoint could be supported by the fact that the dataset is an output of a simulation model instead of realistic cases, and that the simulation tool most probably used a number of existing equations in addition to typical noise.

The inability to interpret the generated principal components is one of the HI-related drawbacks of the PCA and KPCA methods, as was previously mentioned. Thus, effective solutions to cope with this issue should be introduced and substituted. The following paragraphs provide an overview of the proposed methodology's outcomes.

The developed method constructed the following formula after training with 80% of the de-noised dataset (80 turbofan engines):

$$HI = -0.14X_5X_{15} + X_8 - X_9 - X_{10} - X_{14} - 0.2 \quad (16)$$

where X_i denotes the de-noised data from sensor i. Zero weights have been given to the sensors that had no component in the equation, whereas `{-1, +1}` have been applied to the others. The existence of only one multiplication occurring between the de-noised data shows that only one multiplicative neuron with a bias of $e^b = 0.14$ contributes to the additive layer. The bias of the additive neuron is $b = -0.2$. The INN hyperparameters are listed in Table 1. It is worth mentioning that all of the hyperparameters were selected via a grid search technique applied to the subset FD001 only, and then the same values were also applied to the subset FD003. The potential values for each hyperparameter are shown in Table 2. To better explore the impact of two key hyperparameters, λ and α, on the model's performance, the fitness scores are provided in Table 3. The results demonstrate that the model maintains acceptable performance across these hyperparameter settings, thus supporting its stability and reliability.

Since a combination of activation functions converts the additive neuron into a multiplicative one, the computational complexity is similar to a vanilla feed-forward neural network, i.e. O(L×N2), where L is the number of layers and N is the number of neurons on each layer. The created HIs for each sample of the test data are displayed in Fig. 5(bottom left), demonstrating successful performance for all three metrics (Mo, Tr, and Pr). The overall fitness score is 2.9461, as depicted in Table 4, demonstrating that the INN effectively combined the de-noised data to produce a greater score for criteria. Whereas the multiplicative layer has a large number of multiplicative neurons (32) in order to obtain a concise formula, as expected, only one multiplicative neuron—the one that multiplies the features X_5 and X_{15}—contributes to the output after implementing weight regularization with sparsity control.

Given directly unprocessed raw data as input, the suggested model produces the following equation:

$$HI = 0.04\frac{X_1^{0.4}X_2^{0.3}X_6^{0.2}X_7^{0.1}X_{12}^{0.1}}{X_5^{0.2}X_{14}^{0.3}X_{15}^{0.2}} - X_5 + X_8 - X_9 + X_{11} + 0.11 \quad (17)$$

where X_i denotes the raw data from sensor i. Since it is more challenging to achieve an effective equation given raw noisy data than de-noised ones if only the ternary format of the weights is employed, the HI equation contains more terms, as expected. In order to construct (17), some weights of the multiplication layer had to be float values that were rounded to the closest first decimal point. Fig. 5(top left) displays the created HIs for each—test unit. As can be seen in Table 4, the fitness score for the raw data is 2.7407, which is lower than the de-noised form. Once more, the proposed INN was able to effectively combine the raw data to generate a bet-

Table 2 The hyperparameters' spaces for grid search

Alpha (α)	[1.1, 1.3, . . . , 1.9]	Multiplicative Neurons	[1, 16, 32, 64, 128]
Lambda (λ)	[10^{-5}, 10^{-4}, …, 10^{-2}]	Additive Neurons	[1, 16, 32, 64, 128]
Batches	[4, 8, 16]	Learning Rate	[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}]
Epochs	[100, 200, 300]		

Table 3 Impact of hyperparameters of Alpha (α) and Lambda (λ) on the performance of the model based on fitness score

Fitness [raw / de-noised]		Alpha (α)			
		1.2	1.4	1.6	1.8
Lambda (λ)	10^{-5}	2.34 / 2.55	2.51 / 2.77	2.66 / 2.95	2.56 / 2.82
	10^{-4}	2.39 / 2.61	2.48 / 2.75	2.57 / 2.83	2.60 / 2.84
	10^{-3}	2.47 / 2.68	2.49 / 2.76	2.53 / 2.81	2.74 / 2.86
	10^{-2}	2.36 / 2.56	2.34 / 2.58	2.41 / 2.68	2.45 / 2.69

ter HI in terms of criteria and interpretability. We doubled the number of neurons in the multiplicative layer (Table 1) since processing raw data is more complicated. This makes training more complex, but the sparsity control once again eliminates the unneeded weights to still generate a simple formula. The hyperparameters α and λ have to be raised to increase the zeroed weights and emphasize this process more, respectively, so that this doubling of the neurons can be compensated. For comparison, the outputs of the most recent study (a two-stage GP model [40]) are shown in Fig. 5(right), along with the outputs of the proposed approach. It should be emphasized that although the equation produced from the two-stage GP model was only applied to and resulted from de-noised data, we also used the same constructed equation to generate HIs given the raw data in order to make a comparison. Tables 4 and 5 present, respectively, the evaluation metrics scores for the test set and for the whole dataset (100 units). The latter is because the prognostic model, which also needs to be trained on only the training portion, could achieve more accurate RUL predictions for the test portion when the criteria scores for the entire HIs, including the training and test portions, are high, rather than only for the test HIs.

The developed model using the de-noised data has the best fitness score of all (2.95). PCA-based HI has a close fitness score (2.94), but the derived HI equation is complicated to comprehend. The GP model similarly achieves a high score (2.93); however, the authors [40] only took into account the highest-quality inputs (based on the feature extractor in the first stage) in the second stage (which has been dedicated to the feature fusion process). It should be emphasized that a larger INN including more layers and neurons with decreasing α could have produced even higher fitness scores, but with less interpretability and more complex functions. The results show that the developed methodology is superior for the subset FD001 as a result of the highest fitness score and, in the meantime, good interpretability.

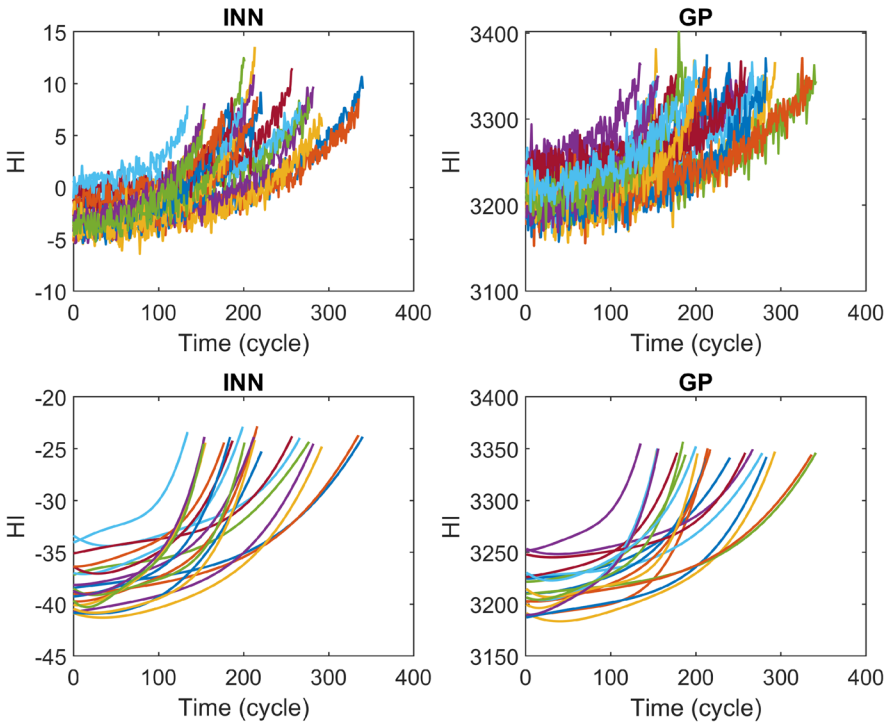


Fig. 5 HIs constructed by the proposed (INN) and two-stage GP (GP) models, utilizing raw (first row) and de-noised (second row) data for 20 test engine units of subset FD001

Table 4 Scores for HI evaluation criteria of PCA, KPCA, GP, and the proposed model (INN), all trained on 80 engine units from subset FD001, calculated considering the 20 test engine units.

		PCA	KPCA	GP	Proposed model	Best Sensor (S 8)
Monotonicity	Raw	0.99	0.96	0.97	0.98	0.96
	Denoised	1.00	1.00	1.00	1.00	1.00
Trendability	Raw	0.92	0.64	0.83	0.92	0.78
	Denoised	0.96	0.95	0.97	0.99	0.97
Prognosability	Raw	0.96	0.71	0.92	0.83	0.87
	Denoised	0.97	0.70	0.97	0.96	0.95
Fitness	Raw	2.87	2.31	2.73	2.74	2.61
	Denoised	2.94	2.65	2.93	2.95	2.91
* "Green color → Red color" equalizes "Best result → Worst result"					*midpoint*	

4.2 Subset FD003

The PCA- and KPCA-based HIs for subset FD003, considering the whole dataset, are shown in Fig. 6. Similar to the subset FD001, sensor 8 has the highest fitness score for raw inputs, 2.56, which has been diminished using the PCA model to 2.29 (-10.55%). This value for de-noised inputs decreased from 2.80 (sensor 8) to 2.47 (-11.79%). In contrast to the subset FD001, the KPCA model provides slightly better HIs compared to the PCA model, with scores of 2.37 and 2.49 for raw and de-noised inputs, respectively. However, they are still less than the best input (sensor 8).

The INN model constructed the following equation after training with 80% of the de-noised dataset:

$$HI = -3.04\frac{X_5X_8X_9X_{10}X_{16}}{X_{14}X_{15}} - 1.51 \tag{18}$$

where X_i denotes the de-noised data from sensor i. In comparison to (16) for the subset FD001, although the equation format has changed, the same sensors are involved, except for sensor 16 (X_{16}) which was the different sensor used in subset FD003. Although the same sensors were included because the objects (engines) are of a special model, the new sensor 16 was also included in the formula, possibly because the subset FD003 contains engines that have two failure modes (rather than one), and thus this sensor likely carries the information related to the failure modes. As can be seen, mainly multiplication neurons contribute to a bias of the additive neuron, which is $b = -1.51$. It should be mentioned that the INN hyperparameters for the subset FD003 are the same as the FD001 (Table 1) since the purpose is also the interpretability rather than purely the high criteria scores. The created HIs for each sample of the test data are displayed in Fig. 7(bottom left), demonstrating successful performance for all three metrics. The overall fitness score is 2.8624, as depicted in Table 6, confirming the INN's performance.

Given directly unprocessed raw data of the subset FD003 as input, the suggested model produces the following formula:

$$HI = -0.024\frac{X_1^{0.2}X_2^{0.2}X_5^{0.1}X_6^{0.3}X_7^{0.2}X_8^{0.2}X_9^{0.3}X_{11}^{0.1}X_{12}^{0.3}X_{14}^{0.3}X_{16}^{0.2}}{X_{14}^{0.1}X_{15}^{0.1}} - 0.1(X_1 + X_2 - X_6) - 3.29 \tag{19}$$

where X_i denotes the raw data from sensor i. Similar to the subset FD001, to construct (19) for the noisy data, some weights of the multiplication layer had to be float values that were rounded to the closest first decimal point. Fig. 7(top left) displays the created HIs for each test unit. As can be seen in Table 6, the fitness score for the raw data is 2.7893, which is lower than the de-noised form. Again, the suggested INN was successful in combining the raw data to produce a better HI in terms of criteria and interpretability. Training becomes more complicated as a result, but the sparsity control once more excludes the extra weights to produce a straightforward equation. In Fig. 7, the outputs of the INN and the two-stage GP model [40] are displayed for comparison. It is important to note that, even though the equation generated by the

Table 5 Scores for HI evaluation criteria of PCA, KPCA, GP, and the proposed model (INN), all trained on 80 engine units from subset FD001, calculated considering both the 80 training and 20 test units.

		PCA	KPCA	GP	Proposed model	Best Sensor (S 8)
Monotonicity	Raw	0.99	0.97	0.98	0.99	0.97
	Denoised	1.00	1.00	1.00	1.00	1.00
Trendability	Raw	0.93	0.60	0.79	0.90	0.74
	Denoised	0.97	0.95	0.97	0.99	0.97
Prognosability	Raw	0.93	0.65	0.90	0.81	0.87
	Denoised	0.97	0.68	0.96	0.96	0.94
Fitness	Raw	2.85	2.22	2.67	2.70	2.58
	Denoised	2.94	2.63	2.93	2.94	2.91
* "Green color → Red color" equalizes "Best result → Worst result"					*midpoint*	

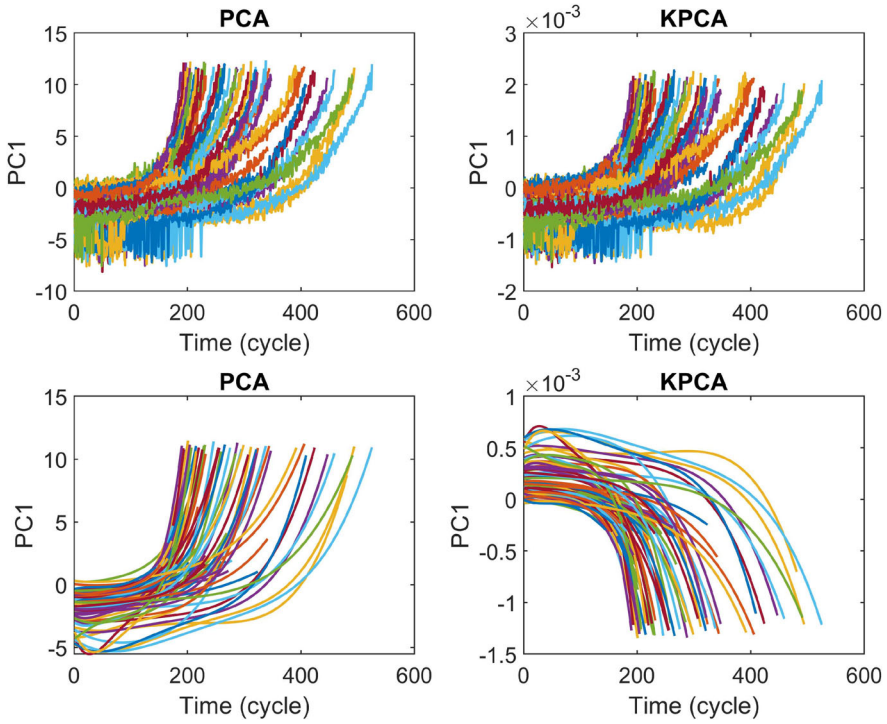


Fig. 6 First principal component of the PCA and KPCA applied to the raw (first row) and de-noised (second row) entire dataset of subset FD003 given 100 engine units

two-stage GP model was the result of applying to only the de-noised data of the subset FD001, we also used the same built equation to obtain HIs for the subset FD003 in order to compare results. Tables 6 and 7 present, respectively, the evaluation metrics scores for the test set and for the whole dataset. Regarding the latter, it should be again emphasized that if the criteria scores for the complete set of HIs (both the training and test) are high, it will increase the accuracy of RUL predictions for the test portion.

The INN model using the de-noised data has the highest fitness score (2.86) compared to the other models. PCA and KPCA models generated HIs with lower fitness scores, and

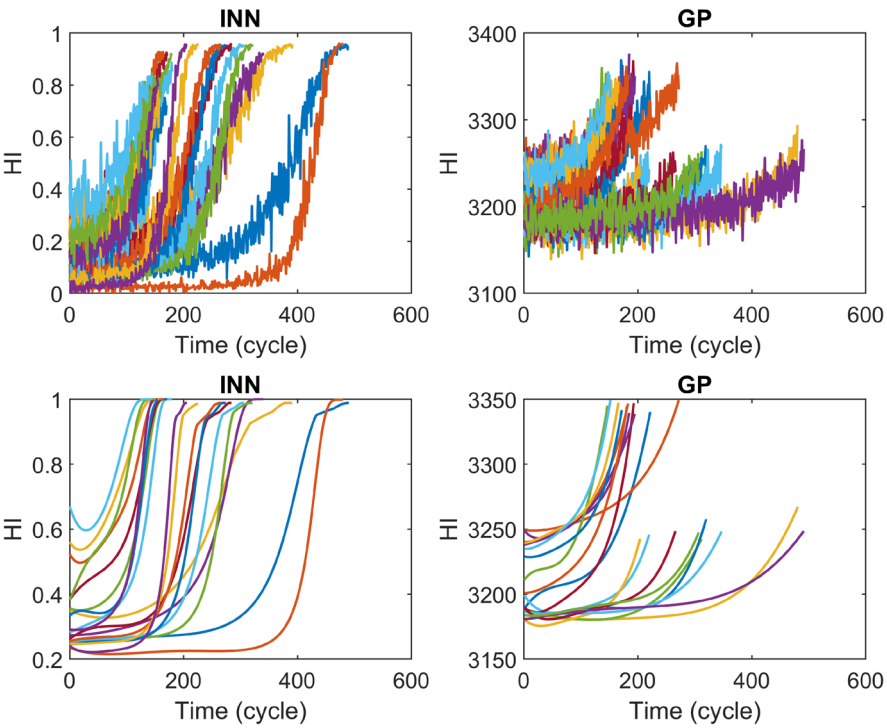


Fig. 7 HIs constructed by the proposed (INN) and two-stage GP (GP) models, utilizing raw (first row) and de-noised (second row) data for 20 test engine units of subset FD003

Table 6 Scores for HI evaluation criteria of PCA, KPCA, GP, and the proposed model (INN), all trained on 80 engine units from subset FD003, calculated considering the 20 test engine units.

		PCA	KPCA	GP	Proposed model	Best Sensor (S 8)
Monotonicity	Raw	0.99	0.98	0.96	0.99	0.97
	Denoised	1.00	1.00	0.99	1.00	1.00
Trendability	Raw	0.90	0.88	0.58	0.91	0.78
	Denoised	0.98	0.98	0.93	0.87	0.97
Prognosability	Raw	0.59	0.63	0.61	0.90	0.92
	Denoised	0.59	0.63	0.60	0.99	0.94
Fitness	Raw	2.47	2.50	2.15	2.80	2.67
	Denoised	2.57	2.60	2.52	2.86	2.91
* "Green color → Red color" equalizes "Best result → Worst result"					*midpoint*	

the extracted HI formula is also complicated to understand. The GP model achieves the lowest score; however, the used equation has been extracted from only the subset FD001. The findings demonstrate that the INN model performs better because of its high fitness score and, concurrently, its high interpretability, i.e., the equation of the built HI can be interpreted and is readable in terms of the inputs. In contrast to the thousands of parameters in typical DL models, the number of parameters in HI equations of INN, given de-noised inputs, is 8 and 9, and given raw inputs, it is 14 and 18, for subsets FD001 and FD003, respectively.

4.3 Health indicator threshold

Since the HI labels were simulated according to a semi-supervised framework with an initial range of [0-1] [44], which can be scaled in any desired range, like [0-10], the end limit was already considered as the threshold of HI for the simulated labels. However, the true threshold of the designed HI for the Time-To-Failure should be selected (or calculated) after the training phase, considering the constructed HIs (predictions) rather than the simulated HIs (targets). With this in mind, the mean value of the constructed HIs at the EoL or at a predefined level based on the intended reliability and safety as well as uncertainty could be selected as the threshold. For example, the range of the simulated labels has been scaled to [0-10] for the subset FD001, while the range for subset FD003 is [0-1] without any scaling, which proves that the employed semi-supervised model is insensitive to the range of labels and only the pattern of targets is important. However, as explained, the constructed HIs are the basis for determining the threshold. With this in mind, for subset FD001, the threshold considering the raw data-based constructed HI is 8.56 and the threshold considering the de-noised data-based constructed HI is -24.04 according to the mean value of the predicted HIs in the training phase. Similarly, these thresholds for subset FD001 considering a higher safety confidence, 5% earlier than the EoL, are 6.09 and -26.56 for the raw data-based and de-nosied data-based constructed HIs, respectively. These threshold values for both subsets in comparison with the testing phase in order to calculate the error between them are shown in Table 8. The error is calculated as ($\text{Threshold}_{Training}$ - Threshold_{Test})/Threshold_{Test}, to consider the test as the basis. The negative errors indicate safer thresholds, while positive ones indicate earlier failures. It should be noted that the true thresholds (at 100% × EoL) for the test units are always greater (later) than the thresholds at 95% × EoL based on the training units, which in turn shows that with a 5% interval, the determined threshold is almost safe.

The INN's ability to directly convolve inputs together is the main reason why it outperforms other compared methods, which lack the ability to multiply inputs, in terms of both higher evaluation scores for HIs and interpretability. Furthermore, sparsity control through the use of discretized (ternary) weights has improved interpretability by making the neural network more compact, resulting in more concise output equations.

Table 7 Scores for HI evaluation criteria of PCA, KPCA, GP, and the proposed model (INN), all trained on 80 engine units from subset FD003, calculated considering both the 80 training and 20 test units.

		PCA	KPCA	GP	Proposed model	Best Sensor (S 8)
Monotonicity	Raw	0.99	0.99	0.96	0.99	0.98
	Denoised	1.00	1.00	0.99	1.00	0.99
Trendability	Raw	0.75	0.75	0.43	0.86	0.68
	Denoised	0.92	0.87	0.73	0.83	0.86
Prognosability	Raw	0.56	0.63	0.60	0.89	0.90
	Denoised	0.56	0.62	0.61	1.00	0.94
Fitness	Raw	2.29	2.37	1.99	2.74	2.56
	Denoised	2.47	2.49	2.33	2.82	2.80
* "Green color → Red color" equalizes "Best result → Worst result"					*midpoint*	

Table 8 Mean value of the constructed HIs at the EoL (100% × EoL) and 5% earlier than the EoL (95% × EoL) to determine the threshold for the Time-To-Failure, which should be based on the training phase

	Inputs	Threshold at 95% × EoL			Threshold at 100% × EoL		
		Training	Test	Error (%)	Training	Test	Error (%)
Subset FD001	Raw data	6.09	5.70	6.84	8.56	8.55	0.12
	De-noised data	-26.56	-26.68	-0.45	-24.04	-23.95	0.38
Subset FD003	Raw data	0.84	0.85	-1.18	0.88	0.90	-2.22
	De-noised data	0.97	0.99	-2.02	0.98	0.99	-1.01

5 Conclusion

Designing an appropriate HI that satisfies the evaluation requirements of monotonicity, trendability, and prognosability for prognostics while also being interpretable for an engineering system or structure in PHM is a challenging task. INN has the potential to combine the CM data and create the intended HI. The majority of ANNs adopt additive neurons, which exclude the ability to multiply the inputs together, probably resulting in a network and equation that are more fundamental. More weighted summation operators will be required instead of multiplication and division (if practical), and the resulting HI product will be more complex. This motivated us to develop a network including a combination of both multiplicative and additive neurons to produce HI in the present work. This work showed the potential of the INNs to achieve ultimate performances by making their prohibitively large equations compact and readable. As such, the HI function has also been simplified by combining the aforementioned multiplicative and additive neurons with the discretized (ternary) weights employing sparsity control. It was shown that even when increasing the number of neurons, the extracted equation is still constructed only by the contributions of a bunch of neurons by controlling the hyperparameters α and λ. The results demonstrated that the proposed methodology is superior based on its combined highest score and interpretability.

To achieve such performances even with the noisy raw data, a small number of weights were excluded from being ternarized and were simply discretized to the first decimal point. Therefore, future research may consider designing a new restricted parameter space that might generalize the discretization technique with more than three points of convergence while maintaining the ternary form as the primary option and the remainder as a secondary option. A technical limitation of the proposed model is capturing the uncertainty of the model, which can in turn improve the model's stability, always resulting in a unique equation. One of the possible solutions could be to differentiate the noise or uncertainty component (which itself can be divided into epistemic and aleatoric uncertainty components) as inferior output functions from the main output function. Similarly, the model can be adapted for components involving multiple faults and different operational conditions via adjusting output functions, or inputs can be clustered based on different failure modes and operational conditions before being fed into the relevant INN model (designed for each scenario) in future work.

Acknowledgements This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 859957 "ENHAnCE, European training Network in intelligent prognostics and Health mAnagement in Composite structurEs ".

Author Contributions **Morteza Moradi:** Conceptualization, Methodology, Software, Validation, Data curation, Formal analysis, Investigation, Visualization, Writing - Original Draft, Writing - Review & Editing. **Panagiotis Komninos:** Methodology, Software, Formal analysis, Investigation, Data curation, Writing - Review & Editing. **Dimitrios Zarouchas:** Writing - Review & Editing, Supervision, Funding acquisition.

Data Availability Dataset used in this study is publicly available.

Declarations

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

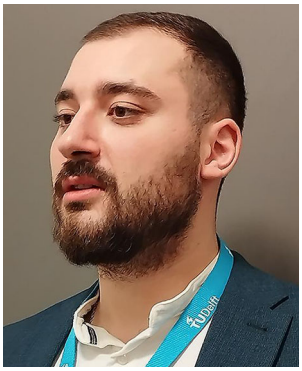
References

1. Guo L, Li N, Jia F, Lei Y, Lin J (2017) A recurrent neural network based health indicator for remaining useful life prediction of bearings. Neurocomputing 240:98–109
2. Soualhi M, Nguyen KT, Medjaher K, Nejjari F, Puig V, Blesa J, Quevedo J, Marlasca F (2023) Dealing with prognostics uncertainties: Combination of direct and recursive remaining useful life estimations. Comput Ind 144:103766
3. Huang R, Xi L, Li X, Liu CR, Qiu H, Lee J (2007) Residual life predictions for ball bearings based on self-organizing map and back propagation neural network methods. Mech Syst Signal Process 21(1):193–207
4. Coble J, Hines JW(2009) Identifying optimal prognostic parameters from data: a genetic algorithms approach. In: Annual Conference of the PHM Society 1
5. Coble J (2010) An Automated Approach for Fusing Data Sources to Identify Optimal Prognostic Parameters. PhD thesis, Dissertation, University of Tennessee Knoxville, TN
6. Niknam SA, Kobza J, Hines JW (2017) Techniques of trend analysis in degradation-based prognostics. Int J Adv Manuf Technol 88:2429–2441

7. Rigamonti M, Baraldi P, Zio E, Roychoudhury I, Goebel K, Poll S (2018) Ensemble of optimized echo state networks for remaining useful life prediction. Neurocomputing 281:121–138
8. Baptista ML, Goebel K, Henriques EM (2022) Relation between prognostics predictor evaluation metrics and local interpretability shap values. Artif Intell 306:103667
9. Hu C, Youn BD, Wang P, Yoon JT (2012) An ensemble approach for robust data-driven prognostics. In: International design engineering technical conferences and computers and information in engineering conference, vol 45028, pp 333–347, American Society of Mechanical Engineers
10. Wen P, Zhao S, Chen S, Li Y (2021) A generalized remaining useful life prediction method for complex systems based on composite health indicator. Reliab Eng & Syst Safe 205:107241
11. Chen D, Qin Y, Qian Q, Wang Y, Liu F (2022) Transfer life prediction of gears by cross-domain health indicator construction and multi-hierarchical long-term memory augmented network. Reliability Engineering & System Safety, pp 108916
12. de Pater I, Mitici M (2023) Developing health indicators and rul prognostics for systems with few failure instances and varying operating conditions using a lstm autoencoder. Eng Appl Artif Intell 117:105582
13. Ni Q, Ji J, Feng K (2022) Data-driven prognostic scheme for bearings based on a novel health indicator and gated recurrent unit network. IEEE Transactions on Industrial Informatics
14. Mitici M, de Pater I, Barros A, Zeng Z (2023) Dynamic predictive maintenance for multiple components using data-driven probabilistic rul prognostics: The case of turbofan engines. Reliability Engineering & System Safety, pp 109199
15. Yan J, He Z, He S (2022) A deep learning framework for sensor-equipped machine health indicator construction and remaining useful life prediction. Comput & Ind Eng 172:108559
16. Molnar C, Casalicchio G, Bischl B (2020) Interpretable machine learning–a brief history, state-of-the-art and challenges. In: Joint european conference on machine learning and knowledge discovery in databases, pp 417–431, Springer
17. Arrieta AB, Díaz-Rodríguez N, Del Ser J, Bennetot A, Tabik S, Barbado A, García S, Gil-López S, Molina D, Benjamins R et al (2020) Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. Inf fusion 58:82–115
18. Sokol K, Flach P (2021) Explainability is in the mind of the beholder: Establishing the foundations of explainable artificial intelligence. arXiv:2112.14466
19. Vollert S, Atzmueller M, Theissler A (2021) Interpretable machine learning: A brief survey from the predictive maintenance perspective. In: 2021 26th IEEE international conference on emerging technologies and factory automation (ETFA), pp 01–08, IEEE
20. Nor AKM, Pedapati SR, Muhammad M, Leiva V (2021) Overview of explainable artificial intelligence for prognostic and health management of industrial assets based on preferred reporting items for systematic reviews and meta-analyses. Sensors 21(23):8020
21. Marcinkevičs R, Vogt JE (2023) Interpretable and explainable machine learning: a methods-centric overview with concrete examples. Wiley Interdisciplinary Rev: Data Mining Knowl Disc 13(3):e1493
22. Cummins L, Sommers A, Ramezani SB, Mittal S, Jabour J, Seale M, Rahimi S (2024) Explainable predictive maintenance: a survey of current methods, challenges and opportunities. IEEE Access
23. Hong CW, Lee C, Lee K, Ko M-S, Kim DE, Hur K (2020) Remaining useful life prognosis for turbofan engine using explainable deep neural networks with dimensionality reduction. Sensors 20(22):6626
24. Khan T, Ahmad K, Khan J, Khan I, Ahmad N (2022) An explainable regression framework for predicting remaining useful life of machines. In: 2022 27th international conference on automation and computing (icac), pp 1–6, IEEE
25. Youness G, Aalah A (2023) An explainable artificial intelligence approach for remaining useful life prediction. Aerospace 10(5):474
26. Baptista M, Mishra M, Henriques E, Prendinger H (2020) Using explainable artificial intelligence to interpret remaininguseful life estimation with gated recurrent unit
27. Protopapadakis G, Apostolidis A, Kalfas AI (2022) Explainable and interpretable ai-assisted remaining useful life estimation for aeroengines. In: Turbo Expo: Power for Land, Sea, and Air 85987, pp V002T05A002, American Society of Mechanical Engineers
28. Solís-Martín D, Galán-Páez J, Borrego-Díaz J (2023) On the soundness of xai in prognostics and health management (phm). Information 14(5):256
29. Lipton ZC (2018) The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. Queue 16(3):31–57
30. Kindermans P-J, Hooker S, Adebayo J, Alber M, Schütt KT, Dähne S, Erhan D, Kim B (2019) The (un) reliability of saliency methods. Interpreting, explaining and visualizing deep learning, Explainable AI, pp 267–280
31. Ghorbani A, Abid A, Zou J (2019) Interpretation of neural networks is fragile. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 3681–3688
32. Moradi M, Gul FC, Zarouchas D (2024) A novel machine learning model to design historical-independent health indicators for composite structures. Compos Part B: Eng 275:111328
33. Veiber L, Allix K, Arslan Y, Bissyandé TF, Klein J (2020) Challenges towards $\{Production - Ready\}$ explainable machine learning. In: 2020 USENIX conference on operational machine learning (OpML 20)
34. Barraza JF, Droguett EL, Martins MR (2024) Scf-net: A sparse counterfactual generation network for interpretable fault diagnosis. Reliab Eng & Syst Safe 250:110285
35. Halford GS, Baker R, McCredden JE, Bain JD (2005) How many variables can humans process? Psych Sci 16(1):70–76
36. Schmidt M, Lipson H (2009) Distilling free-form natural laws from experimental data. Science 324(5923):81–85
37. Udrescu S-M, Tegmark M (2020) Ai feynman: A physics-inspired method for symbolic regression. Sci Adv 6(16):eaay2631
38. Ding P, Qian Q, Wang H, Yao J (2019) A symbolic regression based residual useful life model for slewing bearings. IEEE Access 7:72076–72089
39. Ding P, Jia M, Wang H (2021) A dynamic structure-adaptive symbolic approach for slewing bearings' life prediction under variable working conditions. Struct Health Monit 20(1):273–302
40. Nguyen KT, Medjaher K (2021) An automated health indicator construction methodology for prognostics based on multi-criteria optimization. ISA Trans 113:81–96
41. Moradi M, Komninos P, Benedictus R, Zarouchas D (2022) Interpretable neural network with limited weights for constructing simple and explainable hi using shm data. In: Annual Conference of the PHM Society 14, PHM Society
42. Martius G, Lampert CH (2016) Extrapolation and learning equations. arXiv:1610.02995
43. Deng X, Zhang Z (2022) Sparsity-control ternary weight networks. Neural Netw 145:221–232
44. Moradi M, Broer A, Chiachío J, Benedictus R, Loutas TH, Zarouchas D (2023) Intelligent health indicator construction for prognostics of composite structures utilizing a semi-supervised deep neural network and shm data. Eng App Artif Intell 117:105502
45. Van Engelen JE, Hoos HH (2020) A survey on semi-supervised learning. Machine Learn 109(2):373–440
46. Ramasso E, Saxena A (2014) Review and analysis of algorithmic approaches developed for prognostics on cmapss dataset. In:

Annual conference of the prognostics and health management society 2014
47. Moradi M, Broer A, Chiachío J, Benedictus R, Zarouchas D (2023) Intelligent health indicators based on semi-supervised learning utilizing acoustic emission data. In: European workshop on structural health monitoring, pp 419–428. Springer
48. Durbin R, Rumelhart DE (1989) Product units: A computationally powerful and biologically plausible extension to backpropagation networks. Neural Comput 1(1):133–142
49. Schmitt M (2002) On the complexity of computing and learning with multiplicative neural networks. Neural Comput 14(2):241–301

Morteza Moradi obtained his B.Sc. in mechanical engineering from the University of Kashan, Iran, in 2014, and his M.Sc. (with honors) in aerospace engineering from Iran University of Science and Technology in 2018. Since 2020, he has been a researcher at the Center of Excellence in AI for Structures, Prognostics & Health Management at Delft University of Technology, the Netherlands, where he earned his Ph.D. (cum laude) in aerospace engineering in 2024. He is currently a postdoctoral researcher specializing in AI, signal processing, and information fusion techniques. His research focuses on structural health monitoring, diagnostics, prognostics, and condition-based maintenance in safety-critical industrial sectors, particularly aviation.



Panagiotis Komninos was born in Patras, Greece, in 1997. He received the B.S. degree and the Integrated Master's (Dipl. Ing.) degree in mechanical engineering & aeronautics at the University of Patras, Greece, in 2020. He is currently a Ph.D. candidate in the Center of Excellence in AI for Structures, Prognostics & Health Management, faculty of Aerospace Engineering, Delft University of Technology. He specializes in deep learning methods for facing challenges on varying scientific fields related to industrial applications. His research interests are on developing explainable and hands-on neural networks for supervised, unsupervised, and reinforcement learning applications by narrowing the gap between theoretical and practical AI. His proposed solutions concern interdisciplinary fields including Structural Health Monitoring (SHM), Condition-based Maintenance (CBM), Surrogate modeling, and Decision making.

Dimitrios Zarouchas is an Associate Professor at the Aerospace Engineering Faculty of Delft University of Technology, the Netherlands. He is the Director of the Center of Excellence in AI for Structures, Prognostics & Health Management. The focus of his team is the research, development, and deployment of intelligent systems for enabling real-time diagnostics and prognostics of lightweight structures used in aviation, space, wind energy and naval industries in the context of condition-based (predictive) maintenance.