Interpretable Neural Network with Limited Weights for Constructing Simple and Explainable HI using SHM Data

Morteza Moradi[1,2,*], Panagiotis Komninos[1,2], Rinze Benedictus[1] and Dimitrios Zarouchas[1,2]

[1]*Structural Integrity & Composites Group, Aerospace Engineering Faculty, Delft University of Technology, Kluyverweg 1, 2629 HS Delft, P.O. Box 5058, 2600 GB Delft, The Netherlands*

[2]*Center of Excellence in Artificial Intelligence for structures, prognostics & health management, Aerospace Engineering Faculty, Delft University of Technology, Kluyverweg 1, Delft, 2629 HS, The Netherlands*

M.Moradi-1@tudelft.nl (Corresponding author)

ABSTRACT

Recently, companies all over the world have been focusing on the improvement of autonomous health management systems in order to enhance performance and reduce downtime costs. To achieve this, the remaining useful life predictions have been given remarkable attention. These predictions depend on the proper designing process and the quality of health indicators (HI) generated from structural health monitoring sensors based on prior established multiple prognostic evaluation criteria. Constructing such HIs from noisy sensory data demands powerful models that enable the automatic selection and fusion of features taken from those relevant measurements. Deep learning models are promising to autonomously extract features in scenarios with a huge volume of data without requiring considerable domain expertise. Nonetheless, the features established by artificial neural networks are complicated to comprehend and cannot be regarded as physical system characteristics. In this regard, the goal of this paper is to extend a new model; an interpretable artificial neural network that enables the automatic selection and fusion of features to construct the most appropriate HIs with remarkably fewer parameters. This model consists of additive and multiplicative layers that provide a feature fusion that better reflects the system's physical properties. Additionally, the weights are discretized in two ways: a) using a ternary form with values {-1, 0, 1}, and b) relaxing the aforementioned ternary form by rounding the weights at the first decimal point in the range of [-1, 1]. Both discretization techniques have the ability to softly control the number of parameters that should be ignored. This trick guarantees interpretability for the neural network by extracting simple yet powerful equations representing the constructed HIs. Finally, the model's performance is evaluated and compared with other approaches using a practical case study. The results show that the proposed approach's designed HIs are both interpretable and of high quality according to the criteria of the HI's evaluation.

1. INTRODUCTION

The Health Indicator (HI) is an important index of a structure or engineering system that shows the state of the component's health so that suitable maintenance decisions could be taken. The sensory data collected by structural health monitoring (SHM) techniques can be used to extract the information needed to generate a HI. However, the raw nature of all the initial data produced by SHM methods makes it possible that it is not helpful. The design of HIs for diagnostic and prognostic purposes from often uninformative raw sensor data is indeed a challenging task, but a necessary feature (Galanopoulos, Milanoski, Broer, Zarouchas, & Loutas, 2021). Despite the fact that HI has its own advantages, such as interpretability and a more direct relationship to the component's damage (health) state, it can be imported into a prognostic model to forecast the remaining useful life (RUL). It should be highlighted that higher qualified HI results in more accurate RUL predictions, which improves decision-making strategies.

The efficiency and reliability of HI throughout service life significantly influences the performance of diagnostic and prognostics approaches (Loutas et al., 2019). HIs that could be utilized for diagnostic and prognostic purposes should conform to the HI's evaluation criteria. Three main and well-known prognostic criteria are Monotonicity (Mo), Prognosability (Pr), and Trendability (Tr). Mo reflects a general ascending or descending trend of the HI, while Pr quantifies the distribution of the HI failure values. On the other hand, Tr determines whether degradation trajectories of a specific system/structure/component possess the same underlying pattern (Nick Eleftheroglou, Zarouchas, Loutas, Alderliesten, & Benedictus, 2018). The high Mo, Pr, and Tr scores for a designed HI imply that it is a very suitable index for use in a prediction model. Even a simple prognostic model, such as linear regression, can accurately predict RUL given high criteria scores. Calculating Tr and Pr requires the

Morteza Moradi et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 United States License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

availability of two or more specimens/components, resulting in more than two HIs. If a function can produce HIs with high scores on the above-mentioned prognostic criteria for a collection of similar components, the function can be approved for the future in order to make decisions concerning maintenance tasks.

There are two types of HIs that can be addressed: physical (pHIs) and virtual (vHIs) (Galanopoulos et al., 2021). The first are generated directly from physical measurements, such as static or dynamic strains, ultrasound, temperature, or a combination of these. In fact, the input signals gathered by SHM sensors or their simple combination can sometimes be directly considered as HIs based on the criteria scores, avoiding the need for an additional function or process to analyze and fuse the sensor signals and provide HI, which is relatively rare, especially for complex systems and structures. The latter are typically altered to produce desirable properties such as Mo, Tr, and Pr, which considerably improve prognostic efficiency (Hu, Youn, Wang, & Yoon, 2012; Wen, Zhao, Chen, & Li, 2021). However, another critical aspect of a HI that cannot be assessed by prognostic criteria is its interpretability. Several data-driven algorithms and models have been developed in recent years to provide a good candidate for HI. Yet, the resulting HI functions from data-driven models are almost significantly more complicated than they can be comprehended. Also, the more interpretability of a function, the less overfitting. As a result, the main contribution of the current work is the development of a model to address this challenge.

Common and conventional artificial neural networks (ANNs) use additive neurons, which means that the yields are added together after the inputs are multiplied by weights. As a result, the ability to multiply the inputs together is lost, especially in cases when multiple inputs are significant, such as SHM sensory signals. This mathematical operator may result in a simpler, more comprehensive, and more interpretable function rather than merely considering additive neurons. For the CMAPSS dataset, for example, the HI function proposed by Nguyen and Medjaher (2021) consists only of multiplication and division operators between the features, with no summing operator usage. To replace multiplication and division operators with only summing operators (if feasible), a greater number of weighted summation operators is most likely required, resulting in a more complex, uninterpretable, and incomprehensible HI function.

In the present work, the multiplicative neurons and layers alongside the additive ones will be combined together to make a HI. In order to construct a straightforward yet effective equation for the HI and ensure interpretability for the ANN, 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. The proposed approach is investigated using the turbofan engine degradation simulation dataset released by NASA Ames Prognostics Data Repository, which is extensively applied in the PHM area (Ramasso & Saxena, 2014). The results of PCA, KPCA, and genetic programming (GP) will be compared with the findings. The rest of the paper will be divided into three sections, including Workflow, Results and discussions, and Conclusions.

2. WORKFLOW

First, the overall steps of pre-processing, de-noising, and division of data into training, test, and validation are briefly discussed in the Data section. Then, the HI construction method is presented which includes the additive neuron, the multiplicative neuron, discretized weights, and building the interpretable ANN. Finally, Health indicator evaluation criteria will be presented.

2.1. Data

In the present work, the CMAPSS (Ramasso & Saxena, 2014) dataset (the subset FD001) is used to validate the proposed approach. This dataset was developed by the C-MAPSS tool, which models different deterioration conditions of the fleet of engines from a baseline condition to the final failure in the training data and a duration before the end of life (EoL) in the test data. Except for the first and second columns, which are the ID and deterioration time steps for each engine, and the following three columns, which identify the engine operational parameters, the remaining 21 columns refer to the signals of 21 sensors.

Signals that are constant across all time steps can have a negative effect on data analysis. Thus, at first, data with the same upper and lower bounds is identified and eliminated. In this regard, 6 sensors (1st, 5th, 10th, 16th, 18th, and 19th out of 21) are withdrawn, while 15 remain. The signals are then de-noised to enhance the quality of the subsequent features and HI. In this regard, a regression by a polynomial function of degree four is employed (Nguyen & Medjaher, 2021). Following that, the smoothed signals (features) can be selected as HIs or extracted (feature extraction) and fused (feature fusion) to build a suitable HI. Finally, by importing the designed HIs into the prognostic models, RUL can be predicted.

The subset FD001 in the CMAPSS includes 100 train and test trajectories each. To accurately investigate the prognostic criteria (Mo, Tr, and Pr), however, only the training dataset—which includes data up to EoL—can be used. Therefore, 20% of the training dataset is also taken into consideration and discussed as a validation portion that is not involved during model training.

2.2. HI construction method

Before introducing the proposed methodology for designing an appropriate HI, three popular methods are briefly outlined, and their resulting HIs will be compared with the current approach.

For health and performance indicators, principal component analysis (PCA) can be used to discover lower-dimensional representations of data. Nevertheless, whenever confronted with inhomogeneity and time-varying patterns of system/component degradations, PCA, which is based on a linear transformation of the original data, reveals its shortcomings. As a result, various PCA variants, such as Kernel-PCA (KPCA), PCA-based K-nearest neighbors (KNN), and PCA-based Gaussian mixture models (Thieullen, Ouladsine, & Pinaton, 2012), have been proposed to cope with the challenge of nonlinear data. Yet, the principal components (PCs) generated by the aforementioned PCA-based approaches are not explainable, which could be problematic in some situations. Furthermore, they are typically employed for diagnostic purposes (Ding et al., 2010; Yu, 2011), and hardly ever for prognostics ones that demand further signal processing techniques (Benkedjouh, Medjaher, Zerhouni, & Rechak, 2013; Mosallam, Medjaher, & Zerhouni, 2016). ANN and deep learning (DL) models can be used to autonomously create HIs in scenarios with a huge volume of data without requiring considerable domain expertise. Nonetheless, the features established by DL are complicated to comprehend and cannot be regarded as physical system characteristics. In this regard, a two-stage automated-HI-construction framework based on genetic programming (GP) was proposed, claiming that it requires minimal human involvement and facilitates the generation of interpreted HI (Nguyen & Medjaher, 2021) .

Making an ANN interpretable is not a straightforward task as it depends on the specific domain. Constructing HIs has been recently shown to be effective by just applying some mathematical operators (summation, multiplication) to the extracted features from sensory data (Nguyen & Medjaher, 2021). In this section, we introduce the idea of constructing automatically such mathematical operators inside the ANN to produce simple, yet effective HIs without reducing the high accuracies that deep learning could offer. It should be noted that the ANN is not going to output the equation, but it is the equation itself.

In the present work, multiplicative and additive neurons alongside together are presented with the goal of developing HI, and the results are compared with the outputs of genetic programming (GP) (Nguyen & Medjaher, 2021), PCA, and KPCA models. Semi-supervised learning is used in this study to generate HI by implicitly incorporating HI evaluation metrics (Moradi, Broer, Chiachío, Benedictus, & Zarouchas, 2022). A hypothesized ideal HI function is defined using the prognostic criteria to generate (labels) targets for a supervised ANN to extract the HI function. The optimal function is a quadratic polynomial (HIt = t^2), which is defined by usage time (t) (Moradi et al., 2022). The functions should be normalized using max-min normalization to adopt Pr as a recursive reconstruction method of HI.

2.2.1. Looking Inside the ANN – Additive Neuron

An ANN consists of a collection of connected units called artificial neurons that are grouped together into layers. Signals pass through each layer as inputs. The outputs of one layer become the inputs of the next one. Given some inputs x_K that come after the previous layer,~~ it is possible to apply~~ the ANN's fundamental equation for each neuron separately is:

$$\{\, N_j = \sum_{i=1}^{K} \left(w_{ji}^{l} x_i \right) + \, b^{l} \,\} \qquad (1)$$

where w_{ji}^{l} is the weight corresponding to the link between the $(l{-}1)^{\text{th}}$ layer's i$^{\text{th}}$ neuron to the l^{th} layer's j^{th} neuron, and b^{l} is the bias of the neuron that is added to shift the output of the neuron accordingly. The final output of the neuron is calculated by adding a nonlinearity via an activation function *F(N)* which has the only constraint to be differentiable to the points of interest. While the ANN is being trained, the weights and biases of each neuron, which represent the learnable parameters of the network, are trying to adjust their values by minimizing a loss function (or maximizing an objective function) through backpropagation (Buscema & misuse, 1998). Again, the loss function must be differentiable to the points of interest. This neuron is defined as **additive** since it uses the summation operator over the weighted inputs.

Applicable ANNs for constructing HIs demand thousands or even millions of parameters that make them powerless in terms of interpretability. Indeed, they are called black-box models as it is impossible to interpret the equation that maps the inputs to the outputs. To extract a useful equation that could describe a HI, a small number of neurons and layers should be-involved. We assume that more than two 8-neuron layers could again produce a large, physically unexplainable equation. At first glance, it appears impossible for an ANN to be trained with such a small number of parameters and produce accurate results. Indeed, even with small datasets, it will surely underfit the data. Nevertheless, adding physical properties to the ANN could solve that issue. For the HI construction, physical properties could be simple multiplications and summations among the extracted features, which are formed by the multiplicative and additive layers, respectively, as we will see in the next subsections.

2.2.2. Looking Inside the ANN – Multiplicative Neuron

Forcing the layers to produce such operators demands a modification of the fundamental equation of the neuron (Eq. (1)). Thus, as mentioned in (Durbin & Rumelhart, 1989), instead of having a typical additive neuron, we could have a **multiplicative** neuron by converting the summation step ($\sum_{i=1}^{K} w_{ji} x_i$) into a multiplication step ($\prod_{i=1}^{K} {x_i}^{w_{ji}}$) with the weights as exponents in a product instead of weights in a sum.

This modification demands a logarithmic activation to the inputs before feeding them to Eq. (1) and an exponential activation afterwards. Following these moderations, the equation for converting an additive neuron into multiplicative is as follows:

$$\begin{aligned}\{\, N_j &= e^{\sum_{i=1}^{K} w_{ji}^{l} ln(x_i) + b^l} \\ &= e^{b^l} \cdot e^{\sum_{i=1}^{K} ln(x_i)^{w_{ji}^{l}}} \\ &= e^{b^l} \cdot e^{ln\left(\prod_{i=1}^{K} x_i^{w_{ji}^{l}}\right)} \\ &= e^{b^l} \prod_{i=1}^{K} x_i^{w_{ji}^{l}} \,\}\end{aligned} \qquad (2)$$

In the literature, there is some ambiguity surrounding the term "multiplicative neuron" because it is also used for replacing the summation operator $\sum_{i=1}^{K}\left(w_{ji}x_i\right)$ with a multiplication operator $\prod_{i=1}^{K}\left(w_{ji}x_i\right)$, which slows down the training due to the derivatives that are needed for backpropagation (Schmitt, 2002) and is substantially different from the definition in Eq. (2) that is of our interest. Figure 1 demonstrates the conversion process from additive to multiplicative neurons. Using only these two kinds of activation functions helps the ANN to avoid adding extra nonlinearities that could produce a complicated equation. An important remark here is that forcing these specific activation functions to the neurons limits their scalability as they have the constraint that the inputs should be positive to apply the logarithm. Nevertheless, this is not a pitfall in current work since the inputs could be easily rescaled to the desired range. Finally, since the proposed multiplicative neuron comes naturally from the additive one, the convergence laws of neural networks are satisfied provided that the logarithm exists.

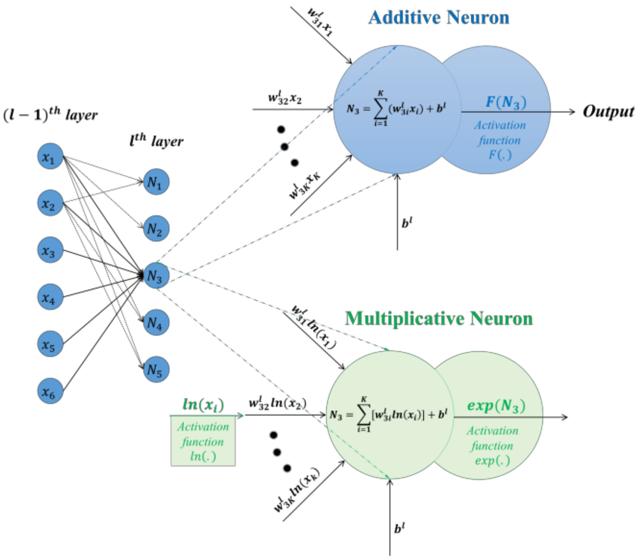


Figure 1. Additive and multiplicative neuron process.

2.2.3. Discretized Weights

Learning with continuous weights is very advantageous as the training is stable and the optimal solution can be found. However, this does not help in constructing compact equations for HIs as the ANN architecture is usually complicated with millions of weights. Even in extreme cases where only few weights are non-zero, having continuous values with many decimal digits guarantees a complicated model. Unfortunately, learning in a continuous space is unavoidable since training an ANN with discrete weights is impossible as the gradients do not exist for back-propagation. A simple solution to having a compact equation could be to round the weights' values to the decimal of interest during testing, but this will negatively alternate the outputs; in many cases, the ANN may become even ineffectual.

Ideally, we wish our weights to be discrete to a specific decimal point or even integers without reducing the accuracy. To address this challenge, ternary weights have been recently introduced (Deng & Zhang, 2022). Rather than rounding the weights to specific decimal digits, the idea is to train an ANN by converging the weights to specific values, in this case to {-1, 0, 1} which also explains this "ternary" term. Certainly, there are cases where we need weights to be somewhere between those three integers. This method does not force all the weights to become integers but a percentage of them, which can be controlled.

Because the full-precision weight space is too large to find an appropriate ternary solution, as mentioned in (Deng & Zhang, 2022), the continuous weight space should be restricted via *tanh(w)*:

$$\{\, w' = tanh(w) \,\} \qquad (3)$$

Now, the weights are restricted to the hyperbolic tangent space ranged in the desired [-1, 1]. This conversion works only by an additional term to the loss function:

$$\{\, L = L_C(y, \hat{y}) + \lambda L_R(w') \} \qquad (4)$$

$$\{\, L_R(w') = \sum_{l=1}^{L} \sum_{w_{ji}} \left[\left(\alpha - tanh^2\left(w_{ji}^{l}\right)\right) tanh^2\left(w_{ji}^{l}\right)\right] \} \qquad (5)$$

$$\{\, L_C(y, \hat{y}) = \frac{1}{n} \sum_{j=1}^{n} (y_j - \hat{y}_j)^2 \,\} \qquad (6)$$

where $L_C(y, \hat{y})$ is the Mean-Squared Loss (MSE) between true and predicted outputs y_j *and* $\hat{y}_j$ respectively over n data points, λ is a regularization constant, L is the number of layers, and α is the shape controller of the loss function $L_R()$. Those λ and α are additional hyperparameters that need tuning for training the ANN. By using the above transformations and loss functions, the gradients exist and are proved to be minimum at *tanh(w) = -1*, *tanh(w) = 0* and *tanh(w) = 1* when *0 < α < 2* (the proof can be found in (Deng & Zhang, 2022)). Another important fact from Eq. (5) is that the percentage of zeros in the trained ternary weights by minimizing $L_R(w')$ is positively related to α and could be monitored to have more or fewer zeroed weights (sparsity control). This is really useful in cases where we have larger ANN architectures and we still wish to have compact equations for HIs by zeroing (increasing α) more weights. The advantage of this modification to the weights and the additional term to the loss function is that the ANN is capable of making accurate predictions by also keeping the weights to their ternary form, and controlling the percentage of them that should be equal to zero.

2.2.4. Building the Interpretable ANN

An ANN from its nature is a function approximator where a complex equation maps the input data to the desired output. Constructing appropriate HIs via an ANN demands millions of parameters, hence, retrieving the equation is impossible. To create an interpretable ANN that could be translated into an expressive and compact equation representing a HI, it is necessary to reduce the number of parameters by retaining its efficiency at high levels. This interpretability is satisfied by combining the discretization of the weights, the sparsity control of the weights, and the utilization of both multiplicative and additive neurons. The weight discretization as well as the sparsity control keep only the important parameters of the ANN which converge to the predefined discrete values during training. Simultaneously, the aforementioned combination of neurons considers the physical properties that silently exist behind the features that construct a HI. Consequently, these tools can now recover the equation behind the ANN that expresses accurately the feature selection and fusion processes, i.e., the HI.

To clarify, many multiplicative/additive neurons within a layer form a multiplicative/additive layer, respectively. The general architecture of the ANN is shown in figure 2. At first, the inputs are fed into a multiplicative layer. Each neuron in the layer is a multiplication between the inputs with different weights and a bias according to Eq. (2). Having many neurons results in different ways of multiplying the inputs. Next, an additive layer with a single neuron sums the outputs of the multiplicative layer to produce the final output. Adding more neurons to the additive layer just makes the ANN more complex and it is very possible to unnecessarily overuse some of the inputs. When a HI's equation is retrieved, terms that refer to a single input are frequently visible rather than a combination of them. For instance, if x_1, x_2, and x_3 are the inputs, we may have an equation $x_1x_2x_3 + x_1$. Using only the outputs of the multiplicative layer to be fed into the additive, it is impossible to produce such an equation. Therefore, the most general architecture is to use the inputs in both additive and multiplicative layers. As such, the outputs of the multiplicative layer are concatenated with the inputs and then are fed into the additive layer.

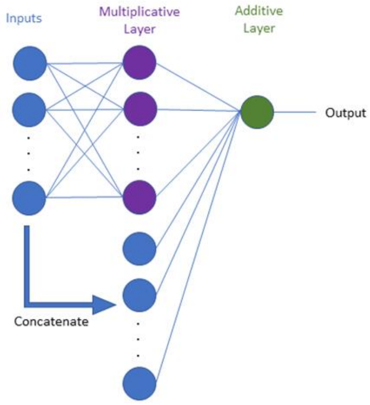


Figure 2. The ANN architecture. The inputs are fed into a multiplicative layer and each neuron applies a multiplication operator. Then, the outputs are concatenated with the inputs and are driven into the additive layer which consists of one neuron and the output is received.

The inputs of the ANN could be either a sequence of raw sensor data, a de-noised format, or some extracted features. The outputs are a sequence of points that form a HI and the trained ANN is the equation for constructing it. Because of varying sequence lengths for each sensor, a preprocessing step is needed before feeding them to the ANN. This step guarantees that the length of the time-series samples will be equal. There are two approaches to achieving this. The simpler one is upsampling by interpolation via adding more data points until every sequence is equal to the largest one. The estimation of those data points depends on the chosen interpolation technique. The second approach is to add pseudo-data points at the end of each sequence until it reaches its maximum length. This could be done via padding with a purposeless value. Then the preprocessed inputs could be fed into the ANN, and the output would be available. The sensitive part of this approach comes during the loss calculation. The padded lengths should be carefully removed to avoid biasing the backpropagation by these pseudo-values. Then, after the parameters are updated, the lengths should be padded again to proceed to the next forward pass. By using this technique, the second approach does not have any approximation step like the first one. However, the training time increases dramatically. In the current case, it was observed that both approaches generate similar results, of which the first one was selected since it is straightforward.

Until now, the equation for constructing the HI was not completely expressive since the weights could have any real value. Using Eqs. (3)-(6) during training, the majority, if not all, of the weights become ternary by moving towards the integers -1, 0, or 1. In practice, values can converge to the desired ones, but they do not always match. In such cases, the values can be safely rounded during test time without reducing the accuracy. As can be seen in the results section, all of the weights become ternary using a de-noised version of the sensor data, but this is not happening when using their raw version. In this last case, a few weights could be anywhere between [-1, 1], which could be simply rounded to the first decimal digit with a negligible reduction in accuracy as long as most of them are in their ternary version. The cause of having some non-ternary weights after training is the messy raw signals. Thus, there is a trade-off between converting the weights to their ternary version and minimizing the L_C loss that depends on the regularization hyperparameter λ. Having a high λ means that we prefer to have more ternary weights (better minimization of L_R) and, consequently, a more compact equation than having optimal model predictions (not an optimal minimization of L_C). Luckily, we aim to build a HI that provides high criteria scores (Mo, Tr, and Pr) rather than merely exact target values, thus focusing more on creating compact equations.

2.3. Health indicator evaluation criteria

A HI must fulfil a set of requirements in order to be accepted a predictive parameter. Mo, Pr, and Tr (Coble & Hines, 2009), the three major criteria for evaluating a HI utilized in this work, are defined as follows:

$$\{\, Mo = \frac{1}{M}\sum_{j=1}^{M} \left| \frac{\sum_{i=1}^{N_j}\sum_{k=1,k>i}^{N_j}(t_k - t_i).\, sgn\big(x(t_k) - x(t_i)\big)}{(N_j - 1)\sum_{i=1}^{N_j}\sum_{k=1,k>i}^{N_j}(t_k - t_i)} \right| \,\} \qquad (7)$$

$$\{\, Tr = \min_{j,k}\left|\rho(x_j, x_k)\right|,\ j,\ k = 1,\ 2,\ \dots, M \,\} \qquad (8)$$

$$\{\, Pr = exp\left(-\frac{\big(std_j x_j(N_j)\big)}{mean_j\big(\left|x_j(1) - x_j(N_j)\right|\big)}\right) \} \qquad (9)$$

where x_j represents the vector of HI on the j^th sample, M represents the number of samples monitored, and N_j denotes the number of observations on the j^th sample. *sgn* and ρ are the sign and Pearson's correlation functions, respectively. The range of the three HI criteria is [0, 1], with 0 representing the lowest and 1 representing the best HI quality. The measurement times for $x_{(t_k)}$ and $x_{(t_i)}$ are denoted by t_k and t_i, accordingly. The covariance is denoted by cov, while the standard deviations of x_j and x_k are denoted by $\sigma_{(x_j)}$ and $\sigma_{(x_k)}$, respectively. To account for all of the following prognostic criteria at once, an objective function called "Fitness" (N Eleftheroglou, 2020) is used:

$$\{\, Fitness = Mo_{HI} + Pr_{HI} + Tr_{HI} \,\} \qquad (10)$$

where the fitness score ranges across [0, 3], with 0 being the worst HI quality and 3 reflecting the optimum.

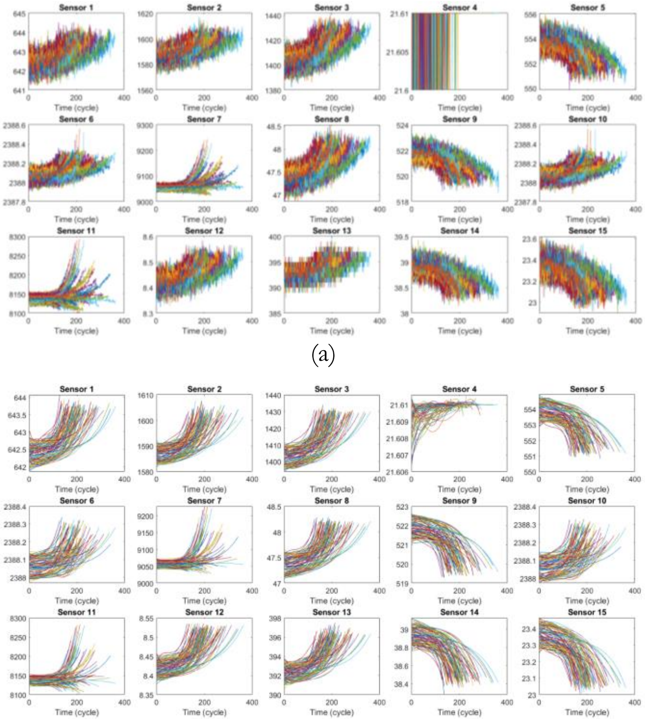
It should be noted that these criteria can only be regarded when all degradation histories up to EoL are available (training dataset). Otherwise, Tr and Pr cannot be measured appropriately (e.g. test dataset).

3. RESULTS AND DISCUSSIONS

In this section, after comparing the raw and de-noised sensor signals in accordance with HI evaluation criteria, HIs produced using the proposed model alongside PCA, KPCA, and GP approaches are evaluated. To assist the ANN in converting its continuous weights into their ternary form, the weights were uniformly initialized in the range [-1, 1]. As it will be discussed later, achieving appropriate results utilizing the raw sensor signals demands the relaxation of the ternary discretization to a softer version, where float discrete values with one decimal point bound into the same range could be used.

3.1. Raw and de-noised data

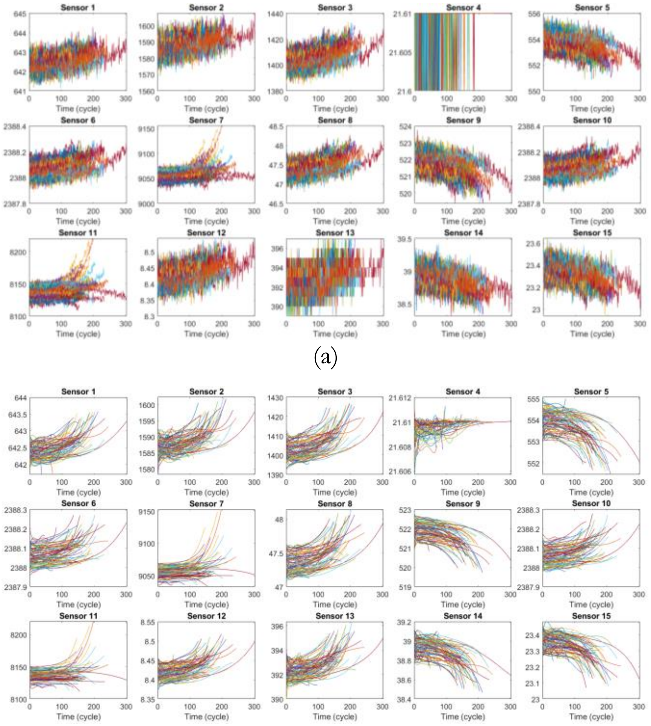
Figures 3 and 4 show the raw and de-noised data for the signals of 15 sensors for the train and test datasets, respectively. The results revealed that the de-noising process adopting 4th-degree polynomial regression was effective. The HI evaluation criteria, consisting of Mo, Pr, and Tr, were also calculated for 15 sensors and reported in Table 1, demonstrating that the de-noising process improves the criteria scores. The test dataset has lower scores, which is reasonable considering that degradation trajectories up to EoL are not accessible. As a result, the scores for 20% of the training dataset as a validation portion, which is not incorporated during model training, are reported in Table 2.



(a)

(b)

Figure 3. (a) Raw and (b) De-noised sensor data for training dataset.



(a)

(b)

Figure 4. (a) Raw and (b) De-noised sensor data for test dataset.

3.2. Health indicators (HIs)

The first principal components of the PCA and KPCA methods can be considered as HI (see figure 5). These results obtained by training the algorithms on the entire training dataset. As can be noticed, standardizing data before to applying the PCA and KPCA algorithms is quite effective for both raw and de-noised data. The de-noising process also leads to an improvement in the fitness scores, but not as much as the standardization process. According to Table 1, the best fitness score for raw inputs is 2.58 (sensor 8), and this score has been enhanced using the PCA algorithm after standardization up to 2.85 (10.47%). This score for de-noised inputs from 2.91 (sensor 8) has been boosted up to 2.94 (1%). However, the KPCA method was unable to enhance the quality of HI with respect to both raw and de-noised inputs, implying that the CMAPSS data has a linear rather than a nonlinear relationship. Thus, for this dataset, a relatively suitable HI can be generated using PCA, and the findings argue that there is no need to develop complicated models for CMAPSS such as deep neural networks (which is what was and is happening nowadays, resulting in tremendous publications). This is also valid for RUL prognosis, as higher HI yields more accurate RUL prediction. This argument could be attributed to the fact that the data is the outcome of a simulation process rather than reality, and several known equations were most likely employed in the simulation process (plus noise).

One of the limitations of the PCA and KPCA algorithms from the standpoint of HI, as previously stated, is the non-interpretability of the generated principal components. As a result, alternative, appropriate approaches to this challenge, such as two-stage GP (Nguyen & Medjaher, 2021), should be developed. The results of the proposed approach are described in the following paragraphs.

The proposed model, which employed the de-noised sensor values from the 4th-degree polynomial regression and was trained on 80% of the training dataset, yielded the following equation:

$$\{ HI = -0.14F_5F_{15} + F_8 - F_9 - F_{10} - F_{14} - 0.2 \} \quad (11)$$

where F_i is the corresponding de-noised sensor i. The sensors that did not contribute to this equation have zeroed weights, whilst the rest have {-1, +1}. Having only one multiplication between the de-noised sensors means that only one multiplicative neuron contributes to the additive layer with a bias $e^b = 0.14$. The additive neuron has a bias of $b = -0.2$. Figure 6b (right) shows the constructed HIs for each sample of the validation set, resulting in high scores for the three criteria (monotonicity, trendability, and prognosability). Indeed, as shown in Table 5, the total criteria score is 2.9461, indicating that the ANN was able to efficiently combine the

Table 1. HI evaluation criteria of raw and de-noised sensor data for both training and test datasets.

			S 1	S 2	S 3	S 4	S 5	S 6	S 7	S 8	S 9	S 10	S 11	S 12	S 13	S 14	S 15
Monotonicity	Raw	Train	0.94	0.94	0.96	0.07	0.96	0.94	0.60	0.97	0.97	0.95	0.56	0.95	0.94	0.95	0.95
		Test	0.80	0.79	0.85	0.06	0.86	0.81	0.38	0.89	0.85	0.78	0.46	0.84	0.74	0.84	0.83
	Den	Train	1.00	0.99	1.00	0.66	1.00	0.99	0.65	1.00	1.00	0.99	0.59	1.00	1.00	1.00	1.00
		Test	0.96	0.93	0.97	0.69	0.97	0.96	0.52	0.98	0.97	0.95	0.50	0.97	0.95	0.97	0.97
Trendability	Raw	Train	3	0.38	0.71	0.00	0.63	0.18	0.01	0.74	0.66	0.17	0.01	0.58	0.43	0.53	0.56
		Test	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Den	Train	0.93	0.88	0.95	0.00	0.96	0.62	0.29	0.97	0.96	0.50	0.02	0.96	0.91	0.97	0.95
		Test	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00
Prognosability	Raw	Train	0.80	0.76	0.87	1.00	0.84	0.65	0.32	0.87	0.85	0.66	0.31	0.84	0.79	0.83	0.81
		Test	0.44	0.38	0.44	0.04	0.39	0.39	0.20	0.40	0.41	0.39	0.17	0.46	0.44	0.47	0.46
	Den	Train	0.92	0.88	0.95	0.73	0.89	0.73	0.33	0.94	0.88	0.73	0.30	0.94	0.90	0.93	0.93
		Test	0.40	0.42	0.41	0.43	0.40	0.36	0.17	0.42	0.42	0.37	0.14	0.45	0.44	0.43	0.45
Fitness	Raw	**Train**	**2.22**	**2.08**	**2.54**	**1.07**	**2.43**	**1.77**	**0.93**	**2.58**	**2.48**	**1.78**	**0.88**	**2.37**	**2.17**	**2.32**	**2.32**
		Test	**1.24**	**1.17**	**1.29**	**0.09**	**1.25**	**1.20**	**0.58**	**1.28**	**1.27**	**1.17**	**0.63**	**1.31**	**1.19**	**1.32**	**1.29**
	Den	**Train**	**2.85**	**2.76**	**2.90**	**1.39**	**2.85**	**2.33**	**1.27**	**2.91**	**2.84**	**2.22**	**0.91**	**2.90**	**2.81**	**2.89**	**2.88**
		Test	**1.36**	**1.36**	**1.38**	**1.12**	**1.37**	**1.32**	**0.69**	**1.40**	**1.39**	**1.32**	**0.65**	**1.42**	**1.41**	**1.40**	**1.42**

* “Green color → Red color” equalizes “Best result → Worst result” 🟥 *midpoint* 🟩

Table 2. HI evaluation criteria of raw and de-noised sensor data for validation (20% of training) datasets.

		S 1	S 2	S 3	S 4	S 5	S 6	S 7	S 8	S 9	S 10	S 11	S 12	S 13	S 14	S 15
Mo	Raw	0.93	0.93	0.96	0.04	0.96	0.93	0.68	0.96	0.96	0.94	0.63	0.94	0.93	0.94	0.94
	Den	0.99	1.00	1.00	0.53	1.00	0.99	0.75	1.00	1.00	0.98	0.66	1.00	1.00	1.00	1.00
Tr	Raw	0.46	0.42	0.70	0.00	0.65	0.14	0.01	0.78	0.71	0.21	0.01	0.60	0.51	0.59	0.58
	Den	0.96	0.95	0.95	0.01	0.99	0.73	0.03	0.96	0.97	0.64	0.09	0.96	0.96	0.96	0.96
Pr	Raw	0.79	0.81	0.88	1.00	0.88	0.72	0.28	0.87	0.89	0.75	0.27	0.80	0.82	0.79	0.83
	Den	0.93	0.91	0.97	0.76	0.89	0.77	0.28	0.95	0.89	0.75	0.26	0.96	0.89	0.93	0.94
Fitness	Raw	**2.18**	**2.16**	**2.53**	**1.04**	**2.49**	**1.79**	**0.96**	**2.61**	**2.56**	**1.90**	**0.92**	**2.34**	**2.26**	**2.33**	**2.36**
	Den	**2.88**	**2.85**	**2.92**	**1.29**	**2.88**	**2.48**	**1.06**	**2.91**	**2.86**	**2.37**	**1.02**	**2.92**	**2.85**	**2.89**	**2.90**

de-noised sensors to generate a higher criteria score than utilizing only the best sensor. Table 3 contains the ANN hyperparameters.

The following is the equation generated by directly applying the proposed model to raw sensor data:

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

where X_i is the corresponding data of sensor i. The HI equation includes more terms when utilizing raw data than using de-noised data, as expected, and it is also difficult to obtain efficient results when solely using the ternary format of the weights. Indeed, some weights of the multiplication layer needed to be float numbers that were rounded to their nearest first decimal digit to produce Eq. (12). The constructed HIs for each sample of the validation set are shown in figure 6a (right). The raw sensor data criterion scores are lower than the de-noised version, as shown in Table 5, with a fitness criteria score of 2.7407. Again, the ANN was able to efficiently fuse the raw sensor data to produce a superior criteria score than if only the best sensor was used. The ANN hyperparameters are stored in Table 4. Because dealing with raw data requires searching within a larger space of weights, we doubled the number of neurons in the multiplicative layer. This adds complexity during training, but thanks to the control of sparsity, we could remove the unwanted weights to produce again a compact equation. Therefore, by doubling the neurons, we had to also increase the hyperparameters α and λ for increasing the zeroed weights and emphasizing more on this process, respectively. In addition to the suggested approach's results, the results of the state-of-the-art work (two-stage GP model (Nguyen & Medjaher, 2021)) are demonstrated in figure 6 (left) for comparison. It should be noted that the equation

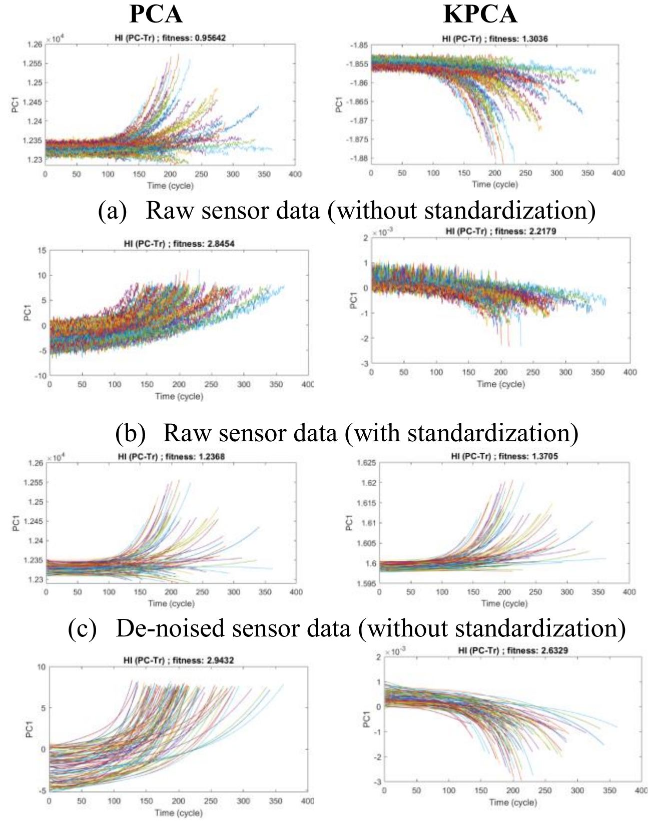
Table 3. ANN's hyperparameters for the de-noised dataset.

Alpha (α)	Lambda (λ)	Batches	Epochs	Multiplicative Neurons	Additive Neurons	Learning Rate
1.6	10^{-5}	4	300	8	1	0.01

Table 4. ANN's hyperparameters for the raw dataset.

Alpha (α)	Lambda (λ)	Batches	Epochs	Multiplicative Neurons	Additive Neurons	Learning Rate
1.8	10^{-3}	4	300	16	1	0.01

derived from the two-stage GP model is based solely on de-noised data, but we also applied it to raw data for comparison. The HI evaluation criteria for the validation and entire training set, respectively, are shown in Tables 5 and 6.



(a) Raw sensor data (without standardization)

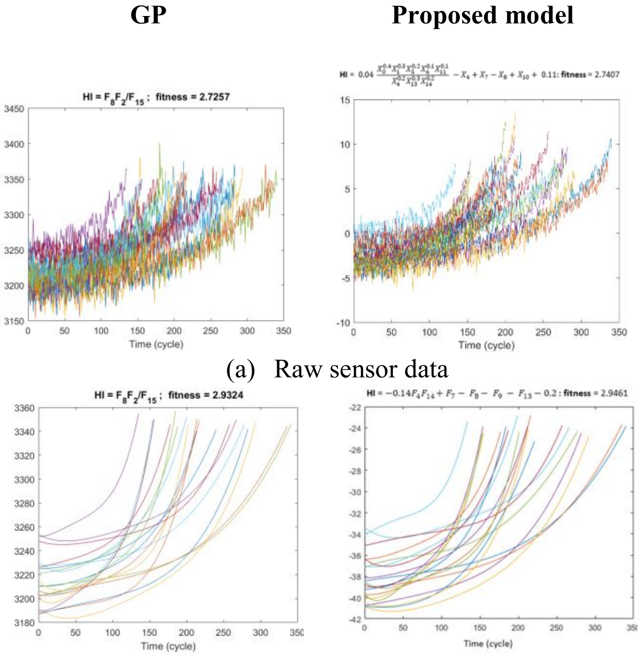
(b) Raw sensor data (with standardization)

(c) De-noised sensor data (without standardization)

(d) De-noised sensor data (with standardization)

Figure 5. The first principal component of the PCA (left column) and KPCA (right column), with and without standardization using zero-mean normalization, for both raw and de-noised training dataset.

The proposed model employing the de-noised data has the highest fitness score (2.95) of all. Despite the fact that PCA's HI has a high close score (2.94), the resulting HI equation is complex to interpret. The GP model also generates a high score (2.93), but the authors did not consider all of the inputs in the second stage (which is responsible for feature fusion task) and instead chose the highest-quality inputs according to the feature extraction in the first stage. It should be noted that increasing the number of neurons and layers in the ANN could have resulted in even higher fitness scores, but with more complicated functions and less interpretability. As a result, the findings demonstrate that the proposed approach is superior based on the highest score as well as interpretability.



(a) Raw sensor data

(b) De-noised sensor data

Figure 6. HIs constructed by (right column) the proposed and (left column) two-stage GP models, with (a) raw and (b) de-noised data using a validation (of 20% of training) set.

Table 5. HI evaluation criteria scores for PCA, KPCA, GP, and the proposed model, which all were trained on 80% of training dataset, calculated from the remaining 20% of training dataset.

		PCA		**KPCA**		**GP**	**Proposed model**	**Best Sensor (S 8)**
		without Standardization	with Standardization	without Standardization	with Standardization			
Mo	Raw	0.68	0.99	0.93	0.96	0.97	0.98	0.96
	Den	0.73	1.00	0.96	1.00	1.00	1.00	1.00
Tr	Raw	0.05	0.92	0.04	0.64	0.83	0.92	0.78
	Den	0.11	0.96	0.33	0.95	0.97	0.99	0.97
Pr	Raw	0.28	0.96	0.36	0.71	0.92	0.83	0.87
	Den	0.27	0.97	0.36	0.70	0.97	0.96	0.95
Fitness	Raw	**1.00**	**2.87**	**1.33**	**2.31**	**2.73**	**2.74**	**2.61**
	Den	**1.11**	**2.94**	**1.66**	**2.65**	**2.93**	**2.95**	**2.91**

Table 6. HI evaluation criteria scores for PCA, KPCA, GP, and the proposed model, which all were trained on the entire training dataset, calculated considering the entire training dataset.

		PCA		**KPCA**		**GP**	**Proposed model**	**Best Sensor (S 8)**
		without Standardization	with Standardization	without Standardization	with Standardization			
Mo	Raw	0.63	0.99	0.91	0.97	0.98	0.99	0.97
	Den	0.66	1.00	0.96	1.00	1.00	1.00	1.00
Tr	Raw	0.00	0.93	0.00	0.60	0.79	0.90	0.74
	Den	0.26	0.97	0.02	0.95	0.97	0.99	0.97
Pr	Raw	0.32	0.93	0.40	0.65	0.90	0.81	0.87
	Den	0.32	0.97	0.39	0.68	0.96	0.96	0.94
Fitness	Raw	**0.96**	**2.85**	**1.30**	**2.22**	**2.67**	**2.70**	**2.58**
	Den	**1.24**	**2.94**	**1.37**	**2.63**	**2.93**	**2.94**	**2.91**

4. CONCLUSIONS

Designing a qualified HI, which matches the evaluation criteria including monotonicity, trendability, and prognosability, and in the meantime being interpretable for an engineering system/structure in PHM is a challenge. ANN can be employed to fuse the SHM data in order to construct the desired HI. Making an ANN interpretable, on the other hand, is a difficult task that varies depending on the domain. In addition, most ANNs use additive neurons, which means that after the inputs are multiplied by weights, the yields are added together. As a result, the ability to multiply the inputs together is lost, perhaps leading to a more basic network and function. If only summing operators are used instead of multiplication and division (if feasible), a larger number of weighted summation operators will be required, resulting in a more complicated HI product. As a result, in the current study, both multiplicative and additive neurons were employed to generate HI. The HI function has also been simplified by using discretized (ternary) weights with sparsity control. Based on the highest score as well as interpretability, the findings show that the proposed approach is superior.

REFERENCES

Benkedjouh, T., Medjaher, K., Zerhouni, N., & Rechak, S. (2013). Remaining useful life estimation based on nonlinear feature reduction and support vector regression. J Engineering Applications of Artificial Intelligence, 26(7), 1751-1760.

Buscema, M. J. S. u., & misuse. (1998). Back propagation neural networks. 33(2), 233-270.

Coble, J., & Hines, J. W. (2009). Identifying optimal prognostic parameters from data: a genetic algorithms approach. Paper presented at the Annual Conference of the PHM Society.

Deng, X., & Zhang, Z. (2022). Sparsity-control ternary weight networks. J Neural Networks, 145, 221-232.

Ding, S., Zhang, P., Ding, E., Naik, A., Deng, P., & Gui, W. (2010). On the application of PCA technique to fault diagnosis. J Tsinghua Science Technology, 15(2), 138-144.

Durbin, R., & Rumelhart, D. E. (1989). Product units: A computationally powerful and biologically plausible extension to backpropagation networks. J Neural computation, 1(1), 133-142.

Eleftheroglou, N. (2020). Adaptive prognostics for remaining useful life of composite structures.

Eleftheroglou, N., Zarouchas, D., Loutas, T., Alderliesten, R., & Benedictus, R. (2018). Structural health monitoring data fusion for in-situ life prognosis of composite structures. J Reliability Engineering System Safety, 178, 40-54.

Galanopoulos, G., Milanoski, D., Broer, A., Zarouchas, D., & Loutas, T. (2021). Health monitoring of aerospace structures utilizing novel health indicators extracted from complex strain and acoustic emission data. J Sensors, 21(17), 5701.

Hu, C., Youn, B. D., Wang, P., & Yoon, J. T. (2012). An ensemble approach for robust data-driven prognostics. Paper presented at the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.

Loutas, T., Eleftheroglou, N., Georgoulas, G., Loukopoulos, P., Mba, D., & Bennett, I. (2019). Valve failure prognostics in reciprocating compressors utilizing temperature measurements, PCA-based data fusion, and probabilistic algorithms. J IEEE Transactions on Industrial Electronics, 67(6), 5022-5029.

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, Cham.

Mosallam, A., Medjaher, K., & Zerhouni, N. (2016). Data-driven prognostic method based on Bayesian approaches for direct remaining useful life prediction. J Journal of Intelligent Manufacturing, 27(5), 1037-1048.

Nguyen, K. T., & Medjaher, K. (2021). An automated health indicator construction methodology for prognostics based on multi-criteria optimization. J ISA transactions, 113, 81-96.

Ramasso, E., & Saxena, A. (2014). Review and analysis of algorithmic approaches developed for prognostics on CMAPSS dataset. Paper presented at the Annual Conference of the Prognostics and Health Management Society 2014.

Schmitt, M. (2002). On the complexity of computing and learning with multiplicative neural networks. J Neural computation, 14(2), 241-301.

Thieullen, A., Ouladsine, M., & Pinaton, J. (2012). A survey of health indicators and data-driven prognosis in semiconductor manufacturing process. J IFAC Proceedings Volumes, 45(20), 19-24.

Wen, P., Zhao, S., Chen, S., & Li, Y. (2021). A generalized remaining useful life prediction method for complex systems based on composite health indicator. J Reliability Engineering System Safety, 205, 107241.

Yu, J. (2011). Local and nonlocal preserving projection for bearing defect classification and performance assessment. J IEEE Transactions on Industrial Electronics, 59(5), 2363-2376.

Morteza Moradi was born in Isfahan, Iran, in 1992. He received the B.Sc. degree in Mechanical Engineering from the University of Kashan, Iran, in 2014 and the M.Sc. degree in Aerospace Engineering from Iran University of Science and Technology (IUST), Tehran, Iran, in 2018. From 2018 to 2020, he was a Research Assistant with Nondestructive Testing and Condition Monitoring Laboratory, IUST. He is currently pursuing his researches as a Ph.D. candidate in the Structural Integrity and Composites (SI&C), Faculty of Aerospace Engineering, Delft University of Technology (TU Delft). His research interests include Non-Destructive Testing (NDT), Structural Health Monitoring (SHM), Thermography, Acoustic Emission, Composite Structures, Finite Element Modeling (FEM), Signal and Image Processing, Artificial Intelligence (AI), and Information Fusion.

Panagiotis Komninos was born in Patras, Greece, in 1997, Greece. He received the B.Sc 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 Structural Integrity and Composites (SI&C), faculty of Aerospace Engineering, Delft University of Technology (TU Delft). He specializes in Deep Learning techniques for solving Mechanical and Aeronautical problems 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. He focuses on challenges from Structural Health Monitoring (SHM), Condition-based Maintenance (CBM), Finite Element Modeling (FEM), and Robotics.