A Neural Network-Evolutionary Computation Framework for Remaining Useful Life Estimation

David Laredo[1], Zhaoyin Chen[1], Oliver Schütze[2], and Jian-Qiao Sun*[1]

[1]School of Mechanical Engineering, University of California, Merced
[2]Department of Computer Science, CINVESTAV, Mexico City, Mexico

Abstract

This paper presents a data-driven framework for estimating the remaining useful life (RUL) of mechanical systems. Two major components make up the framework: a multi-layer perceptron as base regressor and an evolutionary computation algorithm for the tuning of data-related parameters. On the data side, the framework makes use of a strided time window along with a piecewise linear model to estimate the RUL label for each time window within the training sets. Tuning the data-related parameters using the optimization framework here presented allows for the use of simple regressor models, e.g. neural networks with few hidden layers and few neurons at each layer, which can in turn be deployed in environments with very limited resources such as embedded systems. The proposed method is evaluated on the publicly available C-MAPSS dataset. The accuracy of the proposed method is compared against other state-of-the art methods available in the literature and it is shown to perform better while making use of a simpler, compact model.

Index terms— artificial neural networks, moving time window, RUL estimation, C-MAPSS, prognostics, evolutionary algorithms

1. Introduction

Traditionally, maintenance of mechanical systems has been carried out based on scheduling strategies, nevertheless such strategies are often costly and less capable of meeting the increasing demand of efficiency and reliability [1, 2]. Condition based maintenance (CBM) also known as intelligent prognostics and health management (PHM) allows for maintenance based on the current health of the system, thus cutting costs and increasing the reliability of the system [3]. To avoid confusion, here we define prognostics as the estimation of remaining useful component life. The remaining useful life (RUL) of a system can be estimated based on history trajectory data, this approach which we refer here as data-driven can help improve maintenance schedules to avoid engineering failures and to save costs [4].

*jsun3@ucmerced.edu

The existing PHM methods can be grouped into three different categories: model-based [5] , data-driven [6, 7] and hybrid approaches [8, 9].

Model-based approaches attempt to incorporate physical models of the system into the estimation of the RUL. If the system degradation is modeled precisely, model-based approaches usually exhibit better performance than data-driven approaches [10], nevertheless this comes at the expense of having extensive a priori knowledge of the underlying system and having a fine-grained model of such system (which usually involve expensive computations). On the other hand, data-driven approaches tend to use pattern recognition to detect changes in system states. Data-driven approaches are appropriate when the understanding of first principles of system operation is not comprehensive or when the system is sufficiently complex (i.e. jet engines, car engines, complex machinery) such that developing an accurate model is prohibitively expensive. Common disadvantages for the data-driven approaches are that they usually exhibit wider confidence intervals than model-based approaches and that a fair amount of data is required for training. Many data-driven algorithms have been proposed and good prognostics results have been achieved, among the most popular algorithms we can find artificial neural networks (ANN) [11], support vector machine (SVM) [12], Markov hidden chains (MHC) [13].

Over the past few years, data-driven approaches have gained more attention in the PHM community. A number of machine learning techniques, especially neural networks have been successfully applied to the estimate RUL of diverse mechanical systems. ANNs have demonstrated good performance when applied for modeling highly nonlinear, complex, multi-dimensional system without any prior expertise on the system's physical behavior [14]. While the confidence limits for the RUL predictions can not be naturally provided [15], the neural network approaches are promising on prognostic problems.

Neural networks for estimating the RUL of jet engines has been previously explored in [16] where the authors propose a multi-layer perceptron (MLP) coupled with a feature extraction (FE) method and a time window for the generation of the features for the MLP. In the publication the authors demonstrate that a moving window combined with a suitable feature extractior can improve the RUL prediction reported by other similar methods in the literature. In [14] the authors explore an even newer ANN architecture, the so-called convolutional neural networks (CNNs), where they demonstrate that by using a CNN without any pooling layers coupled with a time window the predicted RUL is further improved.

In this paper we propose a novel framework for estimating the RUL of complex mechanical systems. The framework consists of a MLP to estimate the RUL of the system at hand, coupled with an evolutionary algorithm for the fine tuning of data-related parameters, i.e. parameters that define the shape and quality of the features used by the MLP. The publicly available NASA C-MAPSS dataset [17] is used to assess the efficiency and reliability of the proposed framework. This approach allows for simple and small MLPs to obtain better results than those reported in the current literature while using less computing power.

The remainder of this paper is organized as follows: The C-MAPSS dataset is presented in Section 2, then the framework and all of its components are thoroughly reviewed in Section 3. The method is evaluated using the C-MAPSS dataset in Section 4, a comparison with the state-of-the-art is also provided. Finally, our conclusions are presented in Section 5.

2. NASA C-MAPSS Dataset

The NASA C-MAPSS dataset [17] is used to evaluate performance of the proposed method. The C-MAPSS dataset contains simulated data produced using a model based simulation program developed by NASA. The dataset is further divided into 4 subsets composed of multi-variate temporal data obtained from 21 sensors.

For each of the 4 subsets a training and a test set is provided. The training sets include run-to-failure sensor records of multiple aero-engines collected under different operational conditions and fault modes as described in Table 1.

	C-MAPSS			
Dataset	FD001	FD002	FD003	FD004
Train Trajectories	100	260	100	248
Test Trajectories	100	259	100	248
Operating Conditions	1	6	1	6
Fault Modes	1	1	2	2

Table 1: C-MAPSS Dataset details

The data is arranged in an $n \times 26$ matrix where n corresponds to the number of data points in each subset. The first two variables represent the engine and cycle numbers respectively. The following three variables are operational settings which correspond to the operating conditions in Table 1 and have a substantial effect on engine performance. The remaining variables represent the 21 sensors readings that model the engine degradation throughout time.

Each trajectory within the train and test trajectories is assumed to represent the life-cycle of an engine. Each engine is simulated with different initial health conditions (no faults). For each trajectory of an engine the last data entry corresponds to the moment the engine is declared faulty. On the other hand the trajectories within the test sets terminate at some point prior to failure. The aim of the regressor, e.g. MLP, is then to predict the RUL of each engine in the test set. The actual RUL value of each test trajectories are also included in the dataset for verification purposes. Further discussion of the dataset and details on how the data is generated are given in [18].

2.1. Performance evaluation

To evaluate the performance of the proposed approach on the C-MAPSS dataset we make use of two scoring indicators, namely the Root Mean Squared Error (RMSE) which we will address as $e_{rms}(d)$ and a score function proposed in [18] which we refer in this work as RUL Health Score (RHS) and will be addressed as $s_{rh}(d)$.

The scores are defined as follows,

$$e_{rms} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} d_i^2} \tag{1}$$

$$\begin{aligned} s_{rh} &= \frac{1}{N}\sum_{i=1}^{N} s_i \\ s_i &= \begin{cases} e^{-\frac{d_i}{13}} - 1, & d_i < 0 \\ e^{\frac{d_i}{10}} - 1, & d_i \geq 0, \end{cases} \end{aligned} \tag{2}$$

where N is the total number of testing data samples and $d = \hat{y} - y$ is the error between the estimated RUL values $\hat{y}$, and the actual RUL values y for each engine within the test set. It is important to notice that $s_{rh}(d)$ penalizes late predictions more than early predictions since usually late predictions lead to more severe consequences in fields such as aerospace.

3. Framework description

In this section, the proposed ANN-EA based method for prognostics is presented. Our method uses a Multi-Layer Perceptron (MLP) as the main regressor for estimating the RUL of the engines at each subset of the C-MAPSS dataset. For the training sets, the feature vectors are generated by using a strided time window while the labels vector is generated using a constant RUL for the early cycles of the simulation and then linearly decreasing the number of remaining cycles, this is the so-called piecewise linear degradation model [19]. For the test set, a time window is taken from the last sensors readings of the engine and used to predict the RUL of the engine.

The window-size n_w, window-stride n_s, and early-RUL R_e data-related parameters, which for the sake of clarity and formalism in this study are considered as components of a vector $v \in \mathbb{Z}^3$ such that $\mathbf{v} = (n_w, n_s, R_e)$, have a considerable impact on the quality of the predictions made by the regressor. Handpicking the best parameters, i.e. v, for our application is time consuming, furthermore, grid search approaches as the ones used for hyperparameter tuning in neural networks is computationally expensive given the search space inherent to the aforementioned parameters. In this paper, we propose the use of an evolutionary algorithm to fine tune the data-related parameters. The optimization framework here proposed allows for the use of a simple neural network architecture while attaining better results in terms of the quality of the predictions made by the other methods in the current literature.

3.1. The Neural Network Architecture

For this study we propose to use a rather simple MLP architecture and the structure of the network remained consistent for all the four subsets. All the implementations were done in python using the Keras/Tensorflow environment, the source code is publicly available at the git repository `https://github.com/dlaredo/NASA_RUL_-CMAPS-`.

The choice of the network architecture was made using an iterative process; comparing 6 different architectures, training each for 100 iterations using a mini-batch size of 512 and averaging their results over 10 different runs. Two objectives were pursued: that the architecture was compact, e.g. in terms of layers and neurons within each layer and that the performance indicators were minimized.

The process for choosing the network architecture is as follows: First, chose a fixed v, for the next experiment let $v = (30, 1, 140)$. Next, six different ANN architectures are defined, details of the architectures are provided in Appendix A. For each of the six different architectures its performance is assessed using a cross-validation set from subset 1 of C-MAPSS. Table 2 summarizes the results for each tested architectures while Table 3 presents the architecture chosen for the remainder of this work. The chosen architecture provided the best compromise between compactness and performance among the rest of the tested architectures.

	RMSE				RHS			
Tested Architecture	Min.	Max.	Avg.	STD	Min.	Max.	Avg.	STD
Architecture 1	15.51	17.15	16.22	0.49	4.60	7.66	5.98	0.91
Architecture 2	15.24	16.46	15.87	0.47	4.07	6.26	5.29	0.82
Architecture 3	15.77	17.27	16.15	0.45	5.11	8.25	5.93	0.94
Architecture 4	15.13	17.01	15.97	0.47	3.90	7.54	5.65	1.2
Architecture 5	16.39	17.14	16.81	0.23	5.19	6.58	5.98	0.42
Architecture 6	16.42	17.36	16.87	0.30	5.15	7.09	6.12	0.62

Table 2: Results for different architectures for subset 1, 100 epochs

Layer	Shape	Activation	Additional Information
Fully connected	20	ReLU	L1 = 0.1, L2 = 0.2
Fully connected	20	ReLU	L1 = 0.1, L2 = 0.2
Fully connected	1	Linear	L1 = 0.1, L2 = 0.2

Table 3: Proposed Neural Network architecture

3.2. Shaping the data

This section covers the data preprocessing applied to the raw sensors readings in each of the datasets. Although the original datasets contains 21 different sensors readings, some of the sensors do not present much variance or convey redundant information, such sensors are therefore discarded. In the end, only 14 sensor readings out of the 21 are considered for this study, their indices are $\{2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20, 21\}$. The raw measurements are then used to create the strided time windows with window size n_w and window stride n_s. For the training labels, R_e is used at the early stages and then the RUL is linearly decreased. The data is also normalized to be within the range $[-1, 1]$ using the min-max normalization.

$$\hat{x}_i = 2 * \frac{x_i - min(x_i)}{max(x_i) - min(x_i)} - 1, \tag{3}$$

where x_i denotes the m-dimensional vector whose components are all the readings for the *i-th* sensor and $\hat{x}_i$ is the normalized x_i vector.

3.2.1 Time Window Processing

In multivariate time-series based problems such as RUL, more information can be generally obtained from the temporal sequence of data as compared with the multivariate data point at a single time stamp. Let n_w denote the size of the time window, for a time window with a stride $n_s = 1$, all the past sensors values within the time window are collected and put together to form a feature vector $\mathbf{x}$. This approach has successfully been tested in [14, 16] where the authors propose the use of a moving window with values raging from 20 to 30. In this paper we propose not only the use of a moving time window, but also a *strided* time window that updates n_s elements at the time instead of 1. A graphical depiction of the strided time window is shown in Figure 1.

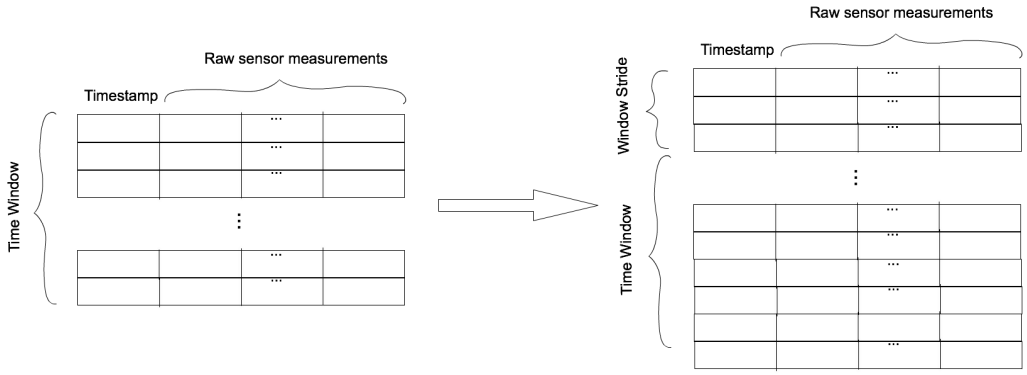

Figure 1: Graphical depiction of the time window used in this framework.

The use of a *strided time window* allows for the regressor to take advantage not only of the previous information available, but also to control the ratio at which the algorithm is fed with new information. With the usual time window approach only one point is updated for every new time window, on the contrary, the strided time window allows for updating n_s points at the time, allowing for the algorithm to catch newer information with fewer iterations, furthermore, the information contained in the strided time window is likely more rich than the one contained in a time window with stride of 1.

3.2.2 Piecewise linear degradation model

Different from common regression problems, the desired output value of the input data is difficult to determine for a RUL problem. It is usually impossible to evaluate the precise health condition and estimate the RUL of the system at each time step without an accurate physics based model. For this popular dataset, a piece-wise linear degradation model has been proposed in [19]. The piece-wise linear degradation model assumes that the engines have a constant RUL label in the early cycles and then the RUL starts degrading linearly until it reaches 0 as shown in Figure 2. The piecewise linear degradation approach is used for this work, in here we denote the value for the RUL at the early stages as R_e.

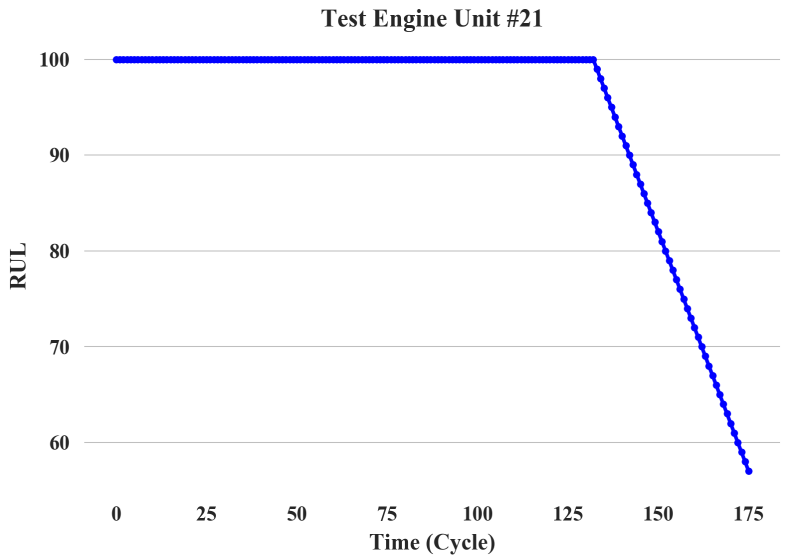

Figure 2: Piecewise linear degradation for RUL.

3.3. Choosing optimal data-related parameters

As mentioned in the previous sections the choice of the data-related parameters v has a large impact on the performance of the regressor, i.e. the MLP. In this section we present a framework for picking the optimal combination of the data-related parameters n_w, n_s and R_e while being computationally efficient.

Recall that $\mathbf{v} = (n_w, n_s, R_e)$, specific to the C-MAPSS dataset are the boundaries of each component, i.e. $n_w \in [1, b]$, $n_s \in [1, 10]$, and $R_e \in [90, 140]$, where all the intervals are integer. The value of b is dependent upon the specific subset, Table 4 presents the different values b can take for each dataset.

	FD001	FD002	FD003	FD004
b	30	20	30	18

Table 4: Allowed values for b per subset

Let also $X(v)$ be the training/cross-val/test sets parametrized by v and used by the MLP to perform the RUL estimation. Finally, let $f(v) = e_{rms}(X(v))$, recall from Equation (1) that $d = \hat{y} - y$ and that $\hat{y}$ depends on $X(v)$, also note that one function evaluation of $f(v)$ implies training the MLP and computing the result of Equation (1). Here we propose to fine tune $\mathbf{v}$, formally speaking

$$\min_{v \in \mathbb{Z}^3} \quad f(v) \tag{4}$$

Given the nature of the problem at hand; namely that no analytical form of the problem is given, gradient information is unavailable and the integer nature of the function variables, an evolutionary algorithm is the natural choice for the optimization process.

3.3.1 Obtaining the true optimal data-related parameters

The size of C-MAPSS dataset and the search space of v allows for an exhaustive search to be performed in order to find the true optimal data-related parameters. We would like to emphasize tough, that although exhaustive search is a possibility for C-MAPSS dataset it is in no way a possibility in a more general setting, therefore the use of the evolutionary algorithm (EA) adopted in this framework. Nevertheless, the possibility to perform exhaustive search on the C-MAPSS dataset can be exploited to demonstrate the accuracy of the chosen EA and of the framework overall. Once again, taking subsets $FD001$ and $FD002$ an exhaustive search is performed to find the true optimal values for v. As was the case when using DE, the MLP is only trained for 20 epochs. Table 5 shows the optimal as well as the worst combinations of data-related parameters and the total number of function evaluations used by the exhaustive search.

Dataset	argmin v	min $f(v)$	argmax v	max $f(v)$	Function evals.
FD001	$[24, 1, 127]$	15.11	$[25, 10, 94]$	85.19	8160
FD002	$[16, 1, 138]$	30.93	$[17, 10, 99]$	59.78	3060

Table 5: Exhaustive search results for subsets FD001 and F002.

3.3.2 Evolutionary algorithms for obtaining the optimal data-related parameters

Evolutionary algorithms (EAs)/meta-heuristics are a family of methods that optimize a problem by iteratively trying to improve a set of candidate solutions with regard to a given measure of quality. The methods do not make any assumptions about the problem, treating it as a black box that merely provides a measure of quality given a candidate solution. Furthermore EAs do not require the gradient of the problem being optimized, making them very suitable for applications such as neural networks. Among the drawbacks of this kind of methods are that they usually require considerable computing effort to converge to a solution.

For this particular application, differential evolution (DE) [20] is chosen as the optimization algorithm. Though in principle any meta-heuristic capable of handling integer variables is suitable for this application, DE has been stablished itself as one of the most reliable, robust and easy to use EAs. Furthermore, a ready to use python implementation is available through the scipy package [21]. Although DE does not have special operators for treating integer variables a very simple modification to the algorithm, consisting on rounding every component of a candidate solution $\mathbf{v}'$ to its nearest integer, is used for this work.

As mentioned earlier, evolutionary algorithms such as DE tend to use several function evaluations for obtaining the optimal solutions, recall that for this application one function evaluations implies retraining the neural network from scratch. This is not a desirable scenario, as obtaining the optimal data-related parameters $\mathbf{v}$ would entail an extensive use of computational power. Instead of running for DE for several iterations and with a large population size we propose to run it just for 30 iterations (generations in the literature of evolutionary computation) and using a population size of 12, which seems reasonable given the size of the search space of $\mathbf{v}$.

Furthermore, during the optimization process the MLP is not trained for 100 epochs, instead the MLP is trained for just 20 epochs, this is done mainly for two reasons: the use of the mini-batch in the training process allows for a speed up in the convergence, therefore it can be assumed that

the algorithm will most likely be very close to its optima after just a couple of iterations, second and most important is the assumption that parameters that lead to lower score values in the early stages of the MLP training process are more likely to provide better performance when trained for the total epochs. Given the similarities between subsets FD001/FD003 and FD002/FD004 we have decided to just tune the for subsets FD001 and FD002. Details for the use of DE in finding the optimum data-related parameters are described in Table 6.

Population Size	Generations	Strategy	MLP epochs
12	30	Best1Bin	20

Table 6: Differential Evolution hyper-parameters.

The optimal data-related parameters for each of the subsets found by DE are shown in Table 7. As can be observed the results obtained by DE are in fact very close to the real optima (Table 5) in both datasets, nevertheless the computational burden is reduced by one order of magnitude when using DE. From the results in Table 7 it can be observed that the maximum allowable time window is always preferred while, on the contrary, small window strides yield better results, for the case of early RUL it can be observed that larger R_e are favored.

Dataset	argmin v	min $f(v)$	Function evals.
FD001	$[24, 1, 129]$	15.24	372
FD002	$[17, 1, 139]$	30.95	372

Table 7: Data-related parameters for each subset obtained with Differential Evolution.

3.4. The ANN-EA RUL estimation Framework

Having described the major building blocks of the proposed method, we now introduce the complete framework in Algorithm 1.

Algorithm 1 ANN-EA RUL estimation Framework

Input: Initial set of data-related parameters $v \in \mathbb{Z}^n$, Raw training/testing data X and training labels y

Output: Optimal set of data-related parameters v^*

1: Choose regressor architecture (ANN, SVM, linear/logistic regression, etc).
2: Define $f(v)$ as in Section3.3.
3: Optimize $f(v)$ using the preferred evolutionary algorithm, i.e. differential evolution, evolutionary strategies, genetic algorithm, etc, using the proposed guidelines from Section 3.3.2.
4: Use v^* to train the regressor for as many epochs as needed.

4. Evaluating the performance of the proposed method

In this section we evaluate the performance of the proposed method. The architecture of the MLP to be used here is described in Table 3 and will be used throughout this section. The MLP was trained 10 times for 200 epochs each and tested in each subset of the C-MAPSS dataset.

For the first experiment, the combinations of window size n_w, window stride n_s and early RUL R_e are presented in Table 8.

Dataset	n_w	n_s	R_e
FD001	24	1	129
FD002	17	1	139
FD003	24	1	129
FD004	17	1	139

Table 8: Data-related parameters for each subset as obtained by DE.

The obtained results for $f(v)$ using the above setting are depicted in Table 9. Notice that the performances obtained for datasets FD001 and FD002 improved, this is due to the fact that this time the MLP was trained for more epochs, thus obtaining better results.

	RMSE				RHS			
Data Subset	min	max	avg	STD	min	max	avg	STD
FD001	14.24	14.57	14.39	0.11	3.25	3.58	3.37	0.11
FD002	28.90	29.23	29.09	0.11	45.99	53.90	50.69	2.17
FD003	14.74	16.18	15.42	0.50	4.36	6.85	5.33	0.95
FD004	33.25	35.10	34.74	0.53	58.52	78.62	74.77	5.88

Table 9: Scores for each dataset using the data-related parameters obtained by DE (Second architecture).

Next, the possibility of using a single set of data-related parameters for all the subsets is explored. For this experiment the n_w is fixed for all of the four datasets, given that the maximum allowable window size for all datasets is 18. Hence, the data-related chosen parameters are $v = (17, 1, 139)$, the results obtained are shown in Table 11.

	RMSE				RHS			
Data Subset	min	max	avg	STD	min	max	avg	STD
FD001	17.58	18.56	17.82	0.29	8.26	9.18	8.59	0.28
FD002	29.48	31.92	30.15	0.75	65.83	104.37	75.73	11.54
FD003	17.35	19.99	18.20	0.72	6.25	16.09	8.27	2.86
FD004	32.91	37.28	34.45	1.41	48.90	77.93	59.60	9.65

Table 10: Scores for each dataset using the single set of data-related parameters.

As can be observed, performance is decreased for subsets FD001/FD003, this indicates that larger window sizes are beneficial for this regression problem. Figures 3 and 4 show a comparison of how the scores are affected for each dataset by changing the data-related parameters to make use of 2 and 1 sets of them.

4.1. Comparison with other approaches

In this section the performance of the proposed method is compared against other state-of-the-art methods. Most of the presented methods in this section have only reported results on the test

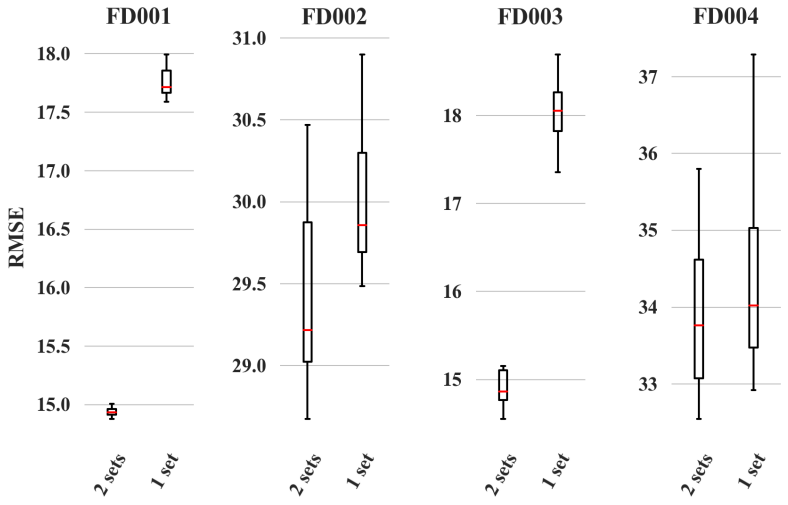

Figure 3: Comparison of RMSE results for different sets of data-related parameters.

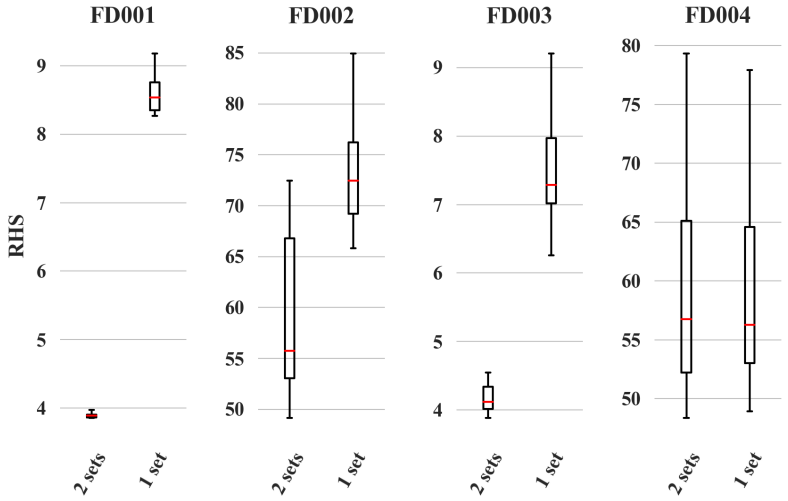

Figure 4: Comparison of RHS results for different sets of data-related parameters.

set FD001 in terms of e_{rms}, the results are displayed in Table 12. The e_{rms} value of the proposed method in Table 12 is the mean value of 10 independent runs. The remainder of the values are identical to those reported in their respective original papers.

Based on the results, the proposed method performs better than the majority of the compared methods when taking into consideration the whole dataset FD001. Two methods come close to the performance of the presented approach in this paper, namely the time window ANN [16] and the Networks Ensemble [24]. While the performance of both methods comes close to the results presented in this paper, the presented approach is computationally more efficient. Furthermore, the framework proposed in here is simple to understand and implement, robust, generic and light-weight, features we believe are important to highlight when comparing the proposed method

Method	e_{rms}
ESN trained by Kalman Filter [22]	63.45
Support Vector Machine Classifier [23]	29.82
Time Window Neural Network [16]	15.16
Multi-objective deep belief networks ensemble [24]	15.04
Deep Convolutional Neural Network [25]	18.45
Proposed method with $n_w = 30$**,** $n_s = 1$ **and** $R_e = 128$	14.87

Table 11: Performance comparisons of the proposed method and the latest related papers on the C-MAPSS dataset.

against other state-of-the-art approaches.

5. Conclusions and future work

This paper presents a novel framework for predicting the RUL of mechanical components. While the method was tested on the jet-engine specific dataset C-MAPSS, the method is general enough so that it can theoretically be applied to other kind of similar systems. The framework makes use of a strided moving time window to generate the training and test sets, a shallow MLP to make the predictions of the RUL and an evolutionary algorithm (DE) which needs to be run just once in order to find the best data-related parameters that optimize the scoring functions used in this study. The results presented in this paper demonstrate that the proposed framework is accurate and computationally efficient, which makes this framework suitable for applications that have limited computational resources such as embedded systems. Furthermore, a comparison with other state-of-the-art methods shown that the proposed method is the best overall performer.

Two major features of the proposed framework are its generality and scalability. While for this paper very specific regressors and evolutionary algorithms were chosen, many other combinations are possible and may be more suitable for different applications. Furthermore, the framework here presented can, in principle, be used for model-construction, i.e. generating the best possible neural network architecture tailored to a specific application. Both issues are to be addressed in future work.

Appendices

A. Tested Neural Network Architectures

Architecture 1

Layer	Shape	Activation	Additional Information
Fully connected	250	ReLU	L1 = 0.1, L2 = 0.2
Fully connected	50	ReLU	L1 = 0.1, L2 = 0.2
Fully connected	1	Linear	L1 = 0.1, L2 = 0.2

Table 12: Proposed Neural Network architecture 4

Architecture 2

Layer	Shape	Activation	Additional Information
Fully connected	100	ReLU	L1 = 0.1, L2 = 0.2
Fully connected	50	ReLU	L1 = 0.1, L2 = 0.2
Fully connected	1	Linear	L1 = 0.1, L2 = 0.2

Table 13: Proposed Neural Network architecture 3

Architecture 3

Layer	Shape	Activation	Additional Information
Fully connected	50	ReLU	L1 = 0.1, L2 = 0.2
Fully connected	20	ReLU	L1 = 0.1, L2 = 0.2
Fully connected	1	Linear	L1 = 0.1, L2 = 0.2

Table 14: Proposed Neural Network architecture 2

Architecture 4

Layer	Shape	Activation	Additional Information
Fully connected	20	ReLU	L1 = 0.1, L2 = 0.2
Fully connected	20	ReLU	L1 = 0.1, L2 = 0.2
Fully connected	1	Linear	L1 = 0.1, L2 = 0.2

Table 15: Proposed Neural Network architecture 1

Architecture 5

Layer	Shape	Activation	Additional Information
Fully connected	20	ReLU	L1 = 0.1, L2 = 0.2
Fully connected	1	Linear	L1 = 0.1, L2 = 0.2

Table 16: Proposed Neural Network architecture 5

Architecture 6

Layer	Shape	Activation	Additional Information
Fully connected	10	ReLU	L1 = 0.1, L2 = 0.2
Fully connected	1	Linear	L1 = 0.1, L2 = 0.2

Table 17: Proposed Neural Network architecture 6

References

[1] N. Z. Gebraeel, M. A. Lawley, R. Liu, and J. K. Ryan. Residual-life distributions from component degradation signals: a bayesian approach. *IEEE Transactions*, 37(6):543–557, 2005.

[2] M.A. Zaidan, A.R. Mills, and R.F. Harrison. Bayesian framework for aerospace gas turbine engine prognostics. In IEEE, editor, *Aerospace Conference*, pages 1–8, 2013.

[3] Z. Zhao, L. Bin, X. Wang, and W. Lu. Remaining useful life prediction of aircraft engine based on degradation pattern learning. *Reliability Engineering & System Safety*, 164:74–83, 2017.

[4] J. Lee, F. Wu, W. Zhao, M. Ghaffari, L. Liao, and D. Siegel. Prognostics and health management design for rotary machinery systems - reviews, methodology and applications. *Mechanical Systems and Signal Processing*, 42(12):314–334, 2014.

[5] W. Yu and H. Kuffi. A new stress-based fatigue life model for ball bearings. *Tribology Transactions*, 44(1):11–18, 2001.

[6] J. Liu and G. Wang. A multi-state predictor with a variable input pattern for system state forecasting. *Mechanical Systems and Signal Processing*, 23(5):1586–1599, 2009.

[7] A. Mosallam, K. Medjaher, and N. Zerhouni. Nonparametric time series modelling for industrial prognostics and health management. *The International Journal of Advanced Manufacturing Technology*, 69(5):1685–1699, 2013.

[8] M. Pecht and Jaai. A prognostics and health management roadmap for information and electronics rich-systems. *Microelectronics Reliability*, 50(3):317–323, 2010.

[9] J. Liu, M. Wang, and Y. Yang. A data-model-fusion prognostic framework for dynamic system state forecasting. *Engineering Applications of Artificial Intelligence*, 25(4):814–823, 2012.

[10] Y. Qian, R. Yan, and R. X. Gao. A multi-time scale approach to remaining useful life prediction in rolling bearing. *Mechanical Systems and Signal Processing*, 83:549–567, 2017.

[11] N. Z. Gebraeel, M. A. Lawley, R. Li, and V. Parmeshwaran. Residual-life distributions from component degradation signals: a neural-network approach. *IEEE Transactions on Industrial Electronics*, 51(3):150–172, 2004.

[12] T. Benkedjouh, K. Medjaher, N. Zerhouni, and S. Rechak. Remaining useful life estimation based on nonlinear feature reduction and support vector regression. *Engineering Applications of Artificial Intelligence*, 26(7):1751–1760, 2013.

[13] M. Dong and D. He. A segmental hidden semi-markov model (hsmm)-based diagnostics and prognostics framework and methodology. *Mechanical Systems and Signal Processing*, 21(5):2248–2266, 2007.

[14] X. Li, Q. Ding, and J. Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering and System Safety*, 172:1–11, 2018.

[15] J. Z. Sikorska, M. Hodkiewicz, and L. Ma. Prognostic modelling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing*, 25(5):1803–1836, 2011.

[16] P. Lim, C. K. Goh, and K. C. Tan. A time window neural networks based framework for remaining useful life estimation. In *Proceedings International Joint Conference on Neural Networks*, pages 1746–1753, 2016.

[17] A. Saxena and K. Goebel. Phm08 challenge data set. [Online] Available at: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/.

[18] A. Saxena, K. Goebel, D. Simon, and N. Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In IEEE, editor, *International Conference On Prognostics and Health Management*, pages 1–9, 2008.

[19] E. Ramasso. Investigating computational geometry for failure prognostics. *International Journal of Prognostics and Health Management*, 5(1):1–18, 2014.

[20] Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, Dec 1997.

[21] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 06/2018].

[22] Y. Peng, H. Wang, J. Wang, D. Liu, and X. Peng. A modified echo state network based remaining useful life estimation approach. In *IEEE Conference on Prognostics and Health Management*, pages 1–7, 2012.

[23] C. Louen, S. X. Ding, and C. Kandler. A new framework for remaining useful life estimation using support vector machine classifier. In *Conference on Control and Fault-Tolerant Systems*, pages 228–233, 2013.

[24] C. Zhang, P. Lim, A.K. Qin, and K.C. Tan. Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE Transactions on Neural Networks and Learning Systems*, 99:1–13, 2016.

[25] G. S. Babu, P. Zhao, and X. Li. Deep convolutional neural network based regression approach for estimation of remaining useful life. In Springer International Publishing, editor, *21st International Conference on Database Systems for Advanced Applications*, pages 214–228, 2016.