Evaluation of the impact of Long Short-Term Memory parameters in RUL prediction for aero engines

P. Manco*, L. Polverino*, R. Abbate*, M. Caterino*, M. Fera*,
M. Rinaldi*, M.A. Turino*, S. D'Ambra*, R. Macchiaroli*

** Dipartimento di Ingegneria Industriale, University of Campania "Luigi Vanvitelli", Via Roma, 9 81031 – Aversa – Italy (pasquale.manco@unicampania.it)*

Abstract: The paper deals with the application of a Long Short-Term Memory (LSTM) to estimate the Remaining Useful Life (RUL) of NASA engines. The engine's data were retrieved from well-known, free available, datasets, in which 21 variables are monitored during the entire engine life, until failure. Thus, the LSTM was identified as the best method to estimate the engine's RUL because it is based on learning from the data series. The LSTM was developed in Python programming language, using a training dataset for the learning phase and a test dataset to evaluate the performance of the algorithm, both provided by NASA. Then, a sensitivity analysis was carried out to evaluate the impact of three parameters on the Mean Squared Error (MSE) of the RUL and the training computational time, namely: i) the window size, i.e. the number of observations to consider to make predictions; ii) the batch size, i.e the number of samples considered for updating the internal model parameters; iii) the Pearson coefficient, used in the pre-processing phase to identify the most useful variables to give as input to the LSTM algorithm. The results highlighted that the window dimension is the most influential parameter among those considered.

Keywords: RUL estimation, Long Short-Term Memory, machine learning, sensitivity analysis.

I. INTRODUCTION

Artificial Intelligence (AI) and in particular machine learning (ML) techniques are considered the most suitable for Predictive Maintenance (PdM) because they can manage high-dimensional processes with a lot of variables (Adhikari et al., 2018). Data from equipment are processed by algorithms to identify some patterns within the acquired signals and to derive the equipment's Remaining Useful Life (RUL), which is a metric for estimating when the monitored equipment will lose its normal operating conditions or will break down (Mathew et al., 2018). RUL estimation focuses on prognostic and health management and is useful to schedule maintenance at the most suitable time (Kang et al., 2021). The macro-steps to obtain an ML model for predictions are data acquisition, dataset preparation, model training, model validation and, at last, prediction (Dalzochio et al., 2020). A large dataset is necessary to train an ML model. Machine learning algorithms are distinguished into supervised (Ren, 2021) and unsupervised (Khanum et al., 2015) depending on whether the metric to be predicted, just like the RUL, is labelled or not-labelled in the training dataset. Labels are the final output in ML. In the context of RUL prediction, labels are failure-related information (Susto et al., 2015). Main learning problems concern classification, regression and clustering (Shreemali et al., 2021). The estimation equipment' RUL is a typical regression problem because the output of the prediction is a number, such as the time or the number of operating cycles to the next fault, rather than categories as in the case of classification problems. Artificial Neural Network (ANN) is an ML technique that can be adopted to estimate the RUL of equipment (Heimes, 2008). An ANN is a data-driven automatic learning model inspired by the biological nervous system (Okoh et al., 2014). Training an ANN aims to find a general law that can describe the link between inputs and outputs of a defined phenomenon and make good predictions about its future developments even with new input data, that are different from those used to train the model (Mahamad et al., 2010). This paper describes the development of a data-driven model to predict the RUL of a turbofan jet engine. The available data set is labelled, meaning the RUL values (outputs) can be always associated with a set of features (inputs). A feature is a property of the phenomenon being observed which can be measured (Jamwal et al., 2021). Moreover, the dataset is made up of temporal series. A Long Short Term Memory (LSTM) deep learning network (Bruneo & de Vita, 2019), which belongs to the class of Recurrent Neural Network (RNN), has been modelled to solve the supervised regression problem.

A. Background and related works

The Feedforward Neural Network (FNN) with backpropagation allowed to improve the accuracy of the prognosis system to predict bearing failures (Mahamad, Saon and Hiyama, 2010). Multi-Layer Perceptron (MLP) Neural Network was used to predict the RUL of turbo engines (Kang et al., 2021). Recurrent Neural Network (RNN) showed to be one of the best approaches to

improve the accuracy of RUL estimation for complex dynamic systems (Heimes, 2008; Liu et al., 2010). Especially the LSTM network provided additional information on the prediction interval of RUL (Liao, Zhang and Liu, 2018). The NASA Commercial Modular Aero Propulsion System Simulation (C-MAPSS) has been taken as an input dataset to estimate the RUL of propulsion engines through LSTM network models (Hsu & Jiang, 2018; Zhao et al., 2019). Models provided better values of root mean squared error (MSE) compared to the performance of MLP, support vector regression and convolutional neural network models. A framework for testing the robustness of LSTM architecture in predicting the RUL has been provided and validated using the C-MAPSS (Sayah et al., 2021). This research belongs to these advances in the field of estimating and improving the prediction accuracy of the RUL through RNN. All modelling steps are shown and a multi-scenario analysis is conducted to optimize the leading parameters of the LSTM deep learning network.

B. Basics of ANNs

An ANN is made up of an input layer, that takes data from the environment and whose nodes (neurons) are representative of the number of features used as inputs, one or more hidden layers and an output layer (Fig.1).

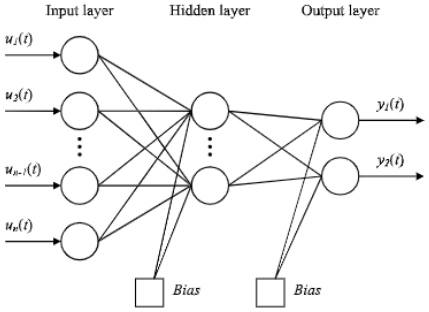

Fig. 1. Architecture of a FFNN (Mahamad et al., 2010)

Each neuron j of the layer *l* can be outlined through the following equation (1):

$$y_j = f\left(\sum_{i=1}^{n} w_{ji}x_i + b_j\right) \; i\epsilon(l-1), j\epsilon l \qquad (1)$$

where w_{ji} is the weight of the link (synapsis) between the neuron *j* and the neuron *i* belonging to the previous layer, x_i is the signal that enters the neuron *j* from the neuron *i*, b_j is the bias of the neuron *j*, *f* is the activation function (Sharma et al., 2020) of the layer l that produces the output y_j according to the biased sum. In a supervised ANN, signals propagate forward into the ANN until the output layer. The output of the ANN is the prediction $\hat{y}$ of the metric, e.g the RUL. It is compared to the actual value of the metric y (the label), that is known. The Mean Squared Error (MSE) between the two values is usually adopted as the cost function to be optimized during the training of the ANN. At each iteration, named epoch, the error backpropagates (Rumelhart et al., 1986). The training aims to find the values of the weights and of the bias that minimizes the cost function. Modelling an ANN is articulated into two main stages: (i) creating the data set and (ii) network implementation. The first stage concerns data collection, the definition and the organization of the features. Implementing the network means choosing the training algorithm and setting hyperparameters, which are the parameters that characterize the network and control the learning process. The main hyperparameters are the number of hidden layers, the number of nodes in each layer, the activation function of the hidden layers and the output layer, the learning rate, the momentum, the training algorithm (Full Batch Gradient Descent, Stochastic Gradient Descent, Mini Batch Gradient descent), the number of epochs, the dataset partition, meaning the percentage of the training set, validation set and test set. When an ANN learns from a dataset, two opposing errors may occur: underfitting and overfitting (Mathworks, 2015). Overfitting is the most frequent one and happens when the model memorizes the dataset rather than learning from it. The result is a model with a poor generalization: it provides good predictions only for scenarios belonging to the training set but it cannot apply the learnt information to new, previously unseen, data. Methods to avoid overfitting and underfitting are well outlined in the literature (Jabbar & Khan, 2015).

C. Recurrent Neural Network

RNN can learn from the sequential dataset (Heimes, 2008). In a sequential dataset, it is not only the contents of the dataset that matter, but also the way data are ordered. This property is very useful for datasets where the memory is crucial, such as temporal series. RNN can replicate the concept of memory by introducing a loop within the hidden layers that links the output of execution to the hidden layer of the next execution; thus, besides influencing the output layer, the output of a node can influence itself in the further temporal step. RNN is trained by using the Backpropagation Through Time (BPTT) algorithm which is an algorithm adapted to RNN from the traditional backpropagation algorithm (Werbos, 1990). The main difference concerns the weights upgrading: the error is not only transmitted backward to the network layers, but also through the temporal executions. The number of multiplications over the network grows significantly (the network has more inputs), especially for very deep networks. For this reason, traditional RNN suffers from the vanishing gradient and exploding gradient problems (Van Houdt et al., 2020), that affect the convergence of the algorithm. The occurrence of these problems can be mitigated using the variant LSTM, which adopts some gates to pass and stored the most important information in a memory cell (Sherstinsky, 2020). In an LSTM network, the training set is not passed to the network all at once, but a window swipes the training set till the end with a constant step of one. The size of the window (WS) is a hyperparameter that the designer must set a priori. This technique is

called sliding window (Yang et al., 2019). It allows to memorize all the information more correctly. Since the size of the windows influences the computational time and the number of acquisitions needed to perform the prediction, several proofs are required to optimize this parameter.

II. MODELLING THE ANN

All phases that led to the creation of the ANN model will be illustrated step by step. They have been outlined in a flowchart (Fig. 2) (Nguyen & Medjaher, 2019).

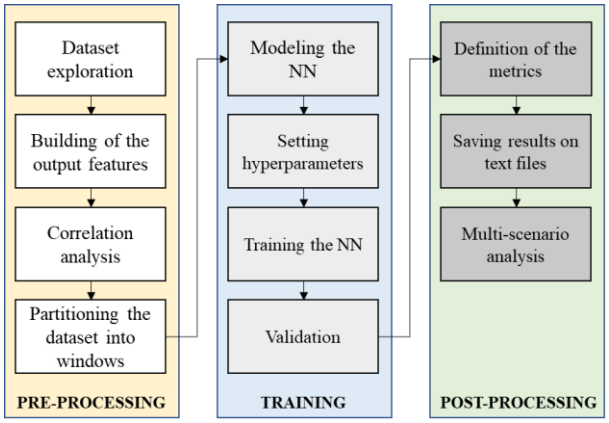

Fig. 2. Flowchart of the ANN modelling process

The tool *Microsoft Excel*® has been adopted to process and elaborate data and information during the pre-processing and post-processing phases, while *Pycharm*® and *Google Colaboratory*® have been used to model, train and validate the ANN in *Python* coding language.

A. *Introduction to the "NASA Turbofan Jet Engine dataset*

Before analysing pre-processing phase, some basic information on the dataset is given. Benchmarking of prognostic algorithms has been challenging due to the limited availability of common datasets suitable for prognostics. In 2010, in an attempt to alleviate this problem, NASA (National Aeronautics and Space Administration) made available the "Turbofan Jet Engine dataset". It is generated by the C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) tool that simulates various degradation scenarios of the fleet of 100 engines of the same type (Ramasso & Saxena, 2014). At the beginning of each scenario, the engine is normally operating. It is degraded until a failure in the training set, which is different for each of the 100 engines. In the test dataset, the acquisitions about the degradation process are stopped before the engine failure. Training and test sets consist of 26 columns that describe the characteristics (features) of the engine. The first and second columns respectively represent the ID and the degradation time steps for every engine. The next three columns characterize the operation modes of the engines (settings) while the final 21 columns correspond to the outputs of 21 sensors, such as temperature and pressure at the inlet of the fan, speed of the fan, fuel-air rate. The C-MAPSS dataset includes 4 subsets: FD001, FD002, FD003 and FD004, which correspond to 4 different cases combining different operating conditions and fault modes. This work focuses on the FD001. The size of the training set dataset is 20631x26 (Fig.3), while the size of the test set is 13096x26. In addition to the training and test datasets, NASA made available the RUL values (100x1) for the engine in the test set. These values represent the number of cycles that pass from the moment the acquisition was stopped to the moment the engine failed.

B. *Pre-processing*

Pre-processing consists of dataset preparation before the ANN starts learning by it. It is developed in three steps:

1. Creating RUL Output Feature.
2. Normalization and Correlation of Data.
3. Dividing Time Series in Time Windows.

machine ID	cycles	setting_1	setting_2	settings_3	s1	. . .	s21
1	1	-0.0007	-0.0004	100	518.67		23.419
1	2	0.0019	-0.0003	100	518.67		23.4236
1	3	-0.0043	0.0003	100	518.67		23.3442
1	4	0.0007	0	100	518.67		23.3739
1	5	-0.0019	-0.0002	100	518.67	. . .	23.4044
		•	•				
		•	•				
		•	•				
100	196	-0.0004	-0.0003	100	518.67	. . .	22.9735
100	197	-0.0016	-0.0005	100	518.67		23.1594
100	198	0.0004	0	100	518.67		22.9333
100	199	-0.0011	0.0003	100	518.67		23.064
100	200	-0.0032	-0.0005	100	518.67		23.0522

Fig. 3. FD001 training dataset

In the training set, the engines reach the point of failure, meaning that the exact cycle in which the engine fails is known. Thus, it is possible to obtain the RUL for each of the 100 engines and each of the cycle steps, through a subtraction operation (2) :

$$RUL_{ij} = max_i - j \qquad (2)$$

where:

- RUL_{ij} is the RUL relating to the jth cycle of the i-th engine;
- max_i is the total number of cycles completed by the i-th engine in its lifetime;
- j is just the corresponding cycle.

Knowing the RUL_{ji} implies that the dataset is labelled. The labelled training set is derived by sorting by column the values of the RUL_{ij} for each engine and each cycle (Fig. 4). Labelling the test dataset requires one more step to obtain the engine lifetime (max_i for test dataset) as the degradation process ends some cycles before the system fails. Therefore, the number of cycles at which acquisitions are stopped must be added to each engine

RUL provided by the "FD001 RUL dataset". The input data are obtained from multiple sensor sources with different ranges of values. Thus, it is necessary to normalize the features' values by their mean and variance to use these heterogeneous data for comparing features and training the LSTM classifier. After data normalization, all features have the same range from zero to one, so it is possible to compare the features. They may not necessarily contain information useful for learning. For example, a feature that remains constant for all rows

machine ID	cycles	setting_1	setting_2	settings_3	s1	. . .	s21	RUL
1	1	-0.0007	-0.0004	100	518.67		23.419	191
1	2	0.0019	-0.0003	100	518.67		23.4236	190
1	3	-0.0043	0.0003	100	518.67		23.3442	189
1	4	0.0007	0	100	518.67		23.3739	188
1	5	-0.0019	-0.0002	100	518.67	. . .	23.4044	187
		.	.					
		.	.					
		.	.					
100	196	-0.0004	-0.0003	100	518.67	. . .	22.9735	4
100	197	-0.0016	-0.0005	100	518.67		23.1594	3
100	198	0.0004	0	100	518.67		22.9333	2
100	199	-0.0011	0.0003	100	518.67		23.064	1
100	200	-0.0032	-0.0005	100	518.67		23.0522	0

Fig. 4. FD001 training dataset with RUL column

of the dataset cannot help the LMST in any way to identify trends or patterns as well as correlated features marginally improve prediction capabilities. Hence, it is crucial, before moving on to creating and training the ANN, to establish the degree of correlation between features to decide whether or not it is right to eliminate one of the two. There are different ways of conducting a statistical correlation analysis between variables, in this case, the one based on the Pearson correlation coefficient (ρ) was chosen. Before deriving the correlations, constant values ("setting3", "s1", "s5", "s10", "s16", "s18" and "s19") have been eliminated because it is reasonable to expect that they cannot contribute during the training. Therefore, the total number of features has gone from 24 to 17. In this study, the Pearson coefficient (P) (Fu et al., 2008) is one of the hyperparameters for the multi-scenario experimental campaign in the post-process phase. Specifically, three levels of correlation were chosen as thresholds (0.7, 0.8 and 1) to establish which features will be considered within the dataset: the case P=1 means that all features are considered for the training; then, variables that are correlated with $P \geq 0.8$ in the first case, and $P \geq 0.7$ in the second case, have been eliminated from the dataset; it needs to be careful to eliminate only one of the two correlated variables. In the first case, when switchin from P=1 to P=0.8, the total number of features is reduced from 17 to 13, while in the second case, when switching from P=08 to P=0.7, the total number of features is reduced from 13 to 10. Fig. 5 shows the heatmap for the level $P \geq 0.7$. It is necessary to divide each time series into many sub-sequences, determined by a scrolling time window, to allow the LSTM ANN to predict the outputs correctly. The window size is the second parameter for the multi-scenario campaign in the post-process phase; four different time windows size has been defined: 10, 25, 50 and 100. To avoid unwanted edge effects, the first two cycles of each engine have been excluded from the subdivision, i.e the first window for each of them will always start from the third cycle. So, once the subdivision for engine #1 has been completed, it passes directly to engine #2 avoiding the common values between them. Therefore, the next time window will start directly from cycle 3 of engine #2. The process ends when the subdivision for all the 100 engines is completed, thus obtaining a certain number of time windows that schematize the entire dataset. It should be emphasized that if an engine is characterized by several acquisitions lower than the window size, no time window can be defined for it. Therefore, engines with this feature will be excluded from the analysis and no predictions can be made for them. The pre-processing phase is finished. Now it is possible to move on to the creation of the ANN.

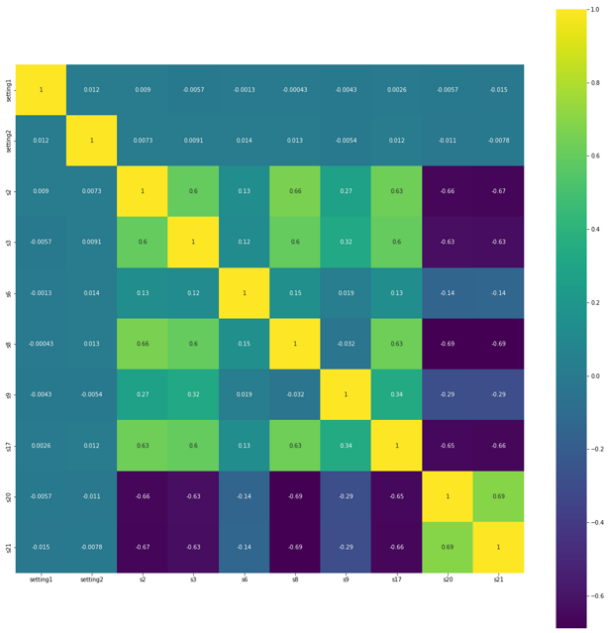

Fig. 5. Heatmap: Pearson correlation $P \geq 0.7$

C. ANN model building

The *Keras* library, which is part of the open-source and wider Tensorflow library, has been used to create the ANN. The network is composed of two types of layers: *dense* and *LSTM*. The dense layer is the simplest type of layer in an ANN, it is typical for the MLP network, while LSTM is needed to avoid the problems of vanishing gradient and exploding gradient. Since the dataset is made up of very long ordered sequences of data with many features, the network has been modelled with three hidden layers. Thus, summing the input and output layers, the total number of layers is five. The summary of the ANN structure for one experiment is shown in Fig. 6. The number of nodes is a trade-off value to keep away the underfitting and overfitting scenarios. It is worth noting that in layer 2 (lstm_1 in Fig. 6) the dropout

technique has been used to avoid overfitting. It consists in switching off a percentage of nodes at each epoch of the training. This percentage, which is called the dropout rate, is itself a hyperparameter that the modeller must set. In this case, it has been set to 0.5, meaning that 50% of the nodes belonging to this layer will be randomly dropped at each epoch.

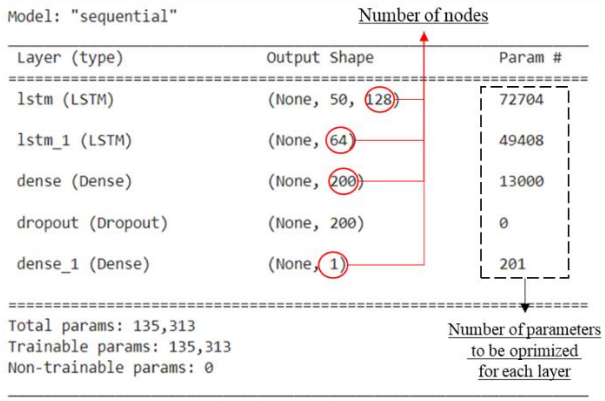

Fig. 6. ANN model for one experiment

The activation functions of the fourth and the output layers, which are both dense layers, are respectively the reLU function and the linear function.

D. Training and validation

The training must be configurated, meaning the modeller have to choose the optimization algorithm and the cost function to be minimized. This operation has been executed with the command *model compile* in Keras. The chosen algorithm is Adam (Jais et al., 2019). It is a variant of the gradient descent for the deep ANN. It adopts a dynamic learning rate for every feature and the momentum. The learning rate has been set to the default value of 0.001. The Mean Square Error (MSE) has been chosen as the cost function. The training is called by the command *model fit*. In this phase, the modeller enters the input (X-train) and output (Y-train) matrices from which the network will learn. The Mini Batch Gradient Descendent algorithm has been selected as a training method. This method executes a descent step for a limited number (a batch) of dataset observations at a time. Thus, it requires setting the batch size (BS), which is usually between 32 and 512. The batch size is another hyperparameter, and it represents also one of the three parameters that will vary in the multi-scenario analysis. Three batch sizes are considered: 128, 256 and 512.; 20% of the training set has been allocated to the validation set, and the number of epochs for the training has been set to 100. The trends of the MSE curves for the training set and the validation set over the training are shown in Fig. 7. As can be seen, the two curves tend to overlap from epoch 80 around a minimum value of the MSE. This outcome means that the training was successful and there will not be any overfitting during the test.

E. Post-processing

Some metrics have to be defined in order to asses quantitatively the performance of the predictive model. An experimental campaign is useful to understand how much parameters influence the selected metrics. Two metrics have been chosen: the MSE and computation time (CT). The formula (3) for the MSE is:

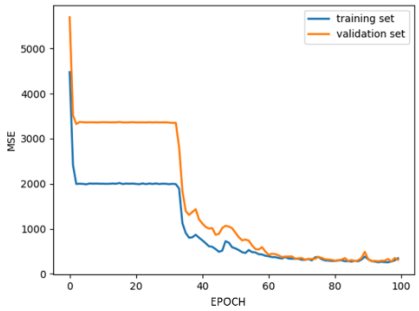

Fig. 7. MSE during the learning process

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad (3)$$

where y_i is the actual value of RUL for the window j, $\hat{y}_i$ is the predicted value of RUL for the window j and n is the total number of windows in the dataset. Moreover, it is crucial to consider the computation time since the time has a cost value in the engineering field; if the computation time is too long for the dynamic of the monitored equipment, decision-makers cannot use the prediction to schedule proper maintenance activities. Results are saved in a text file by using the library *numpy*. For each experiment, the command saves a matrix that indicates the window number, the engine number, the actual RUL and the predicted RUL. Training time is clocked and saved. The multi-scenario analysis consists in executing a series of simulations by varying some parameters in turn (TABLE I). The parameters are: the batch size (BS), the Pearson index (P) and the window size (WS). The scenarios are indicated with the triplet (BS,P,WS). Experiments are carried out keeping fix two parameters and by varying the third.

TABLE I
VALUES OF PARAMETERS IN THE MULTI-SCENARIO ANALYSIS

Parameter	Level 1	Level 2	Level 3	Level 4
Batch size	128	256	512	
Pearson	0,7	0,8	1	
Window size	10	25	50	100

The multi-scenario analysis consists in executing a series of simulations by varying some parameters in turn (TABLE I). The parameters are: the batch size (BS), the Pearson index (P) and the window size (WS). The scenarios are indicated with the triplet (BS,P,WS).

Experiments are carried out keeping fix two parameters and by varying the third. Since the starting weights of the network are initialized randomly, they have been blocked to the same values for each simulation to obtain results best possible comparable.

III. Results Analysis and Discussion

A correlation analysis showed a strong linear correlation between the WS and the two metrics. More precisely, the WS increases as CT linearly increases and MSE linearly decrease. A larger window involves the network makes a more precise prediction because it is based on more acquisitions. Indeed, the time needed to process larger windows is longer. The others parameters do not have a linear correlation with the metrics (Fig.8). This finding underlines that the WS is a very influential parameter, but it does not exclude that the two others are not, because they could have a not-linear correlation with the metrics. Consequently, MSE and CT have been plotted in three scenarios, one for every BS, by using the WS as an independent variable and P as a parameter (Fig. 9).

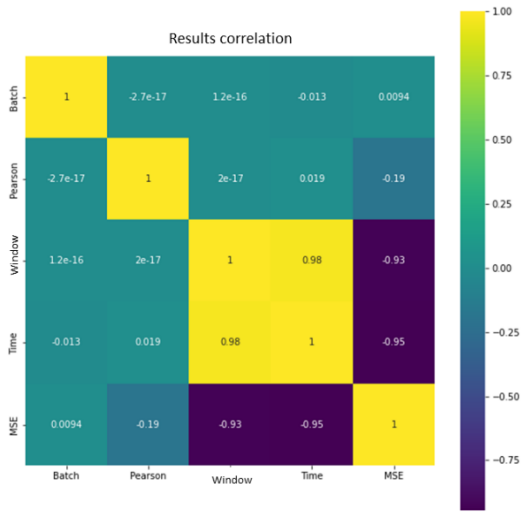

Fig. 8. Correlation between parameters and MSE

Increasing the P implies a reduction of the MSE for the same WS. A higher P means the network is trained with a greater number of features, meaning more information. The only exceptions to this trend are the scenarios (128,0.8,50) and (128,1,100). TC is not affected by Pearson change. The only exception occurs in the scenario (512,1,100). Curves do not show a noticeable change when the BS increases, especially for high values of the BS. This aspect confirms that it impacts little on the accuracy of the prediction. Hence, choosing a higher P results in the lowest MSE without impacting on the TC. Since the WS is the pivoting parameter to design the ANN, it is interesting to derive the percentage variation of MSE and TC between the levels of the WS to understand when it is convenient to expand the WS. The goal is to reduce the MSE as much as possible while keeping TC low. Looking at TABLE II, the most convenient transitions are 25-50 and 50-100, for which a good improvement in accuracy does not correspond to an excessive increase of the TC. Considerations about the optimal values of the design parameters, just like the WS, are crucial, but also other factors must be considered in an industrial environment. The TC varies from 5 min to 35 min. This temporal gap could have more or less large economic implications depending on the company.

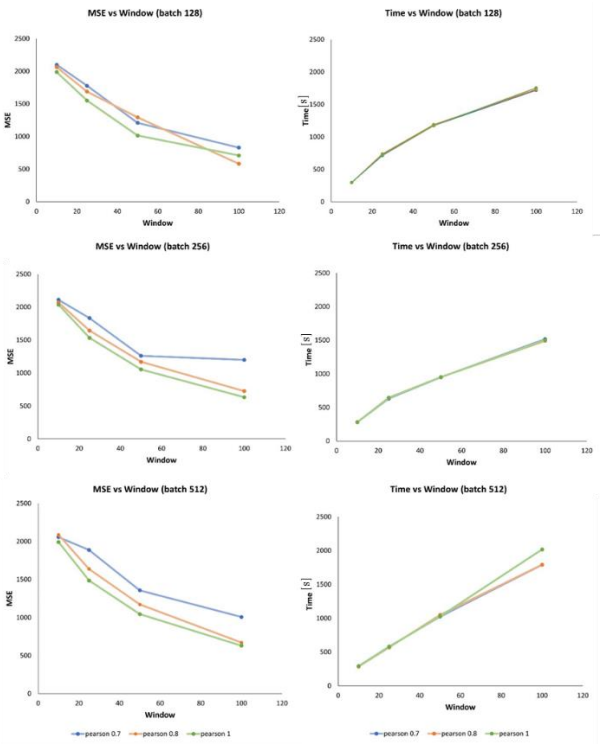

Fig. 9. MSE varying BS, P and WS

TABLE II
Percentage Variation of MSE and TC

BS	P	Transition	MSE [%]	TC [%]
128	1	10-25	-22	+143
128	1	25-50	-35	+62
128	1	50-100	-30	+49
256	1	10-25	-25	+129
256	1	25-50	-31	+48
256	1	50-100	-40	+58
512	1	10-25	-25	+97
512	1	25-50	-30	+78
512	1	50-100	-40	+96

IV. CONCLUSION

This work describes systematically how to model and train an LSTM neural network to predict the RUL of a system subjected to continuous degradation for performing PdM. The NASA dataset for the turbofan engine was used to train and test the model. Before the training, the dataset had to be prepared. Specifically, output features (labels) for the training set and the test set have been derived at first. Then, linear correlation analysis was necessary to find correlated features. Since data are sequential, the sliding window algorithm was used to split the dataset into sub-sequences. An LSTM neural network has been trained through a variant of the backpropagation algorithm and validated by using the early stopping criterion. The script has been written in Python and executed in *Google Colab* and *Pycharm*. A sensitivity analysis with 36 experiments was carried out to assess the impact of the parameters BS, P and WS on the accuracy and performance of the ANN. The MSE and the computation time have been chosen as evaluation metrics. Experiments showed that:

- The WS is the most influential parameter. Increasing the window size from the minimum to the maximum value allows to increase the MSE to 71%. However, the computational time also increases by about 80%.
- Increasing Pearson the MSE decreases without a corresponding increment in TC. Thus, this evidence suggests not to exclude any features from the dataset.
- The BS is the least significant parameter.

This study proved that experiments are the better way to improve the performance of an ANN for the prediction of the RUL of equipment. Further studies will be conducted by implementing a more sophisticated ANN architecture, where "convolutional" layers are added to LSTM and dense layers in a deep neural network.

REFERENCES

[1] Adhikari, P., Rao, H. G., & Buderath, Dipl.-I. M. (2018). Machine Learning based Data Driven Diagnostics & Prognostics Framework for Aircraft Predictive Maintenance. 10th International Symposium on NDT in Aerospace, October 24-26, 2018, Dresden, Germany, Ml.

[2] Bruneo, D., & de Vita, F. (2019). On the use of LSTM networks for predictive maintenance in smart industries. Proceedings - IEEE SMARTCOMP 2019.

[3] Dalzochio, J., Kunst, R., Pignaton, E., Binotto, A., Sanyal, S., Favilla, J., & Barbosa, J. (2020). Machine learning and reasoning for predictive maintenance in Industry 4.0: Current status and challenges. In Computers in Industry (Vol. 123).

[4] Fu, Y., Yan, S., & Huang, T. S. (2008). Correlation metric for generalized feature extraction. IEEE T Pattern Anal, 30(12).

[5] Heimes, F. O. (2008). Recurrent neural networks for remaining useful life estimation. 2008 International Conference on Prognostics and Health Management, PHM.

[6] Hsu, C. S., & Jiang, J. R. (2018). Remaining useful life estimation using long short-term memory deep learning. Proceedings of 4th IEEE ICASI 2018.

[7] Jabbar, H. K., & Khan, R. Z. (2015). Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning

[8] Jais, I. K. M., Ismail, A. R., & Nisa, S. Q. (2019). Adam Optimization Algorithm for Wide and Deep Neural Network. Knowledge Engineering and Data Science, 2(1).

[9] Jamwal, A., Agrawal, R., Sharma, M., Kumar, A., Kumar, V., & Garza-Reyes, J. A. A. (2021). Machine learning applications for sustainable manufacturing: a bibliometric-based review for future research. Journal of Enterprise Information Management.

[10] Kang, Z., Catal, C., & Tekinerdogan, B. (2021). Remaining useful life (Rul) prediction of equipment in production lines using artificial neural networks. Sensors (Switzerland), 21(3).

[11] Khanum, M., Mahboob, T., Imtiaz, W., Abdul Ghafoor, H., & Sehar, R. (2015). A Survey on Unsupervised Machine Learning Algorithms for Automation, Classification and Maintenance. International Journal of Computer Applications, 119(13).

[12] Liao, Y., Zhang, L., & Liu, C. (2018). Uncertainty Prediction of Remaining Useful Life Using Long Short-Term Memory Network Based on Bootstrap Method. IEEE, ICPHM.

[13] Liu, J., Saxena, A., Goebel, K., Saha, B., & Wang, W. (2010). An adaptive recurrent neural network for remaining useful life prediction of lithium-ion batteries, PHM 2010.

[14] Mahamad, A. K., Saon, S., & Hiyama, T. (2010). Predicting remaining useful life of rotating machinery based artificial neural network. Comput Math Appl, 60(4).

[15] Mathew, V., Toby, T., Singh, V., Rao, B. M., & Kumar, M. G. (2018). Prediction of Remaining Useful Lifetime (RUL) of turbofan engine using machine learning. IEEE ICCS 2017,

[16] Mathworks. (2015). Improve Neural Network Generalization and Avoid Overfitting. Matlab R2015.

[17] Nguyen, K. T. P., & Medjaher, K. (2019). A new dynamic predictive maintenance framework using deep learning for failure prognostics. Reliab Eng Syst Safe, 188.

[18] Okoh, C., Roy, R., Mehnen, J., & Redding, L. (2014). Overview of Remaining Useful Life prediction techniques in Through-life Engineering Services. Procedia CIRP, 16.

[19] Ramasso, E., & Saxena, A. (2014). Review and analysis of algorithmic approaches developed for prognostics on CMAPSS dataset. PHM 2014.

[20] Ren, Y. (2021). Optimizing Predictive Maintenance with Machine Learning for Reliability Improvement. ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering, 7(3).

[21] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088).

[22] Sayah, M., Guebli, D., al Masry, Z., & Zerhouni, N. (2021). Robustness testing framework for RUL prediction Deep LSTM networks. ISA Transactions, 113.

[23] Sharma, S., Sharma, S., & Athaiya, A. (2020). Activation functions in neural networks. International Journal of Engineering Applied Sciences and Technology, 04(12).

[24] Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. Physica D: Nonlinear Phenomena, 404.

[25] Shreemali, J., Malviya, L., Paliwal, P., Chakrabarti, P., Poddar, S., Jindal, B., & Chaubisa, H. (2021). Comparing performance of multiple classifiers for regression and classification machine learning problems using structured datasets. Materials Today.

[26] Susto, G. A., Schirru, A., Pampuri, S., McLoone, S., & Beghi, A. (2015). Machine learning for predictive maintenance: A multiple classifier approach. IEEE T Ind Inform, 11(3).

[27] van Houdt, G., Mosquera, C., & Nápoles, G. (2020). A review on the long short-term memory model. Artif Intel Rev, 53(8).

[28] Werbos, P. J. (1990). Backpropagation Through Time: What It Does and How to Do It. Proceedings of the IEEE, 78(10).

[29] Yang, J., Guo, Y., & Zhao, W. (2019). Long short-term memory neural network based fault detection and isolation for electro-mechanical actuators. Neurocomputing, 360.

[30] Zhang, A., Wang, H., Li, S., Cui, Y., Liu, Z., Yang, G., & Hu, J. (2018). Transfer learning with deep recurrent neural networks for remaining useful life estimation. Applied Sciences (Switzerland), 8(12).

[31] Zhao, S., Zhang, Y., Wang, S., Zhou, B., & Cheng, C. (2019). A recurrent neural network approach for remaining useful life prediction utilizing a novel trend features construction method. Measurement: Journal Int Measu Confe, 146.