Multi-Head CNN-LSTM with Prediction Error Analysis for Remaining Useful Life Prediction

Hyunho Mo*†, Federico Lucca†, Jonni Malacarne†, Giovanni Iacca*
*University of Trento
†BlueTensor S.r.l.
Trento, Italy
{hyunho.mo, giovanni.iacca}@unitn.it, {federico, jonni}@bluetensor.ai

***Abstract*—Predicting accurate remaining useful life (RUL) of components plays a crucial role in making optimal decision for maintenance management. As sensor technology develops, multiple sensors are used to collect information for monitoring the condition of components. Deep learning architectures, such as convolutional neural network (CNN) and long short term memory (LSTM), can be considered as a successful end-to-end framework to predict RUL from the multivariate time series collected by those sensors. For that, we employ an architecture combining the parallel branch of CNN in series with LSTM which is referred to as multi-head CNN-LSTM. Furthermore, we propose a combination of the network with time series prediction error analysis (PEA). The prediction errors on the entire time series are estimated by recursive least squares (RLS) and single exponential smoothing (SES) respectively. We analyze each of the two sequences of prediction errors with the exponentially weighted moving average (EWMA) and combine them with the Fisher's method. Finally, the output of the PEA is fed into the multi-head CNN-LSTM network as the additional input. We evaluate the performance of our method on the widely used C-MAPSS dataset. The experimental results suggest that using the PEA improves the performance of the deep learning-based RUL prediction model. Compared to other methods in recent literature, the proposed method achieves the state-of-the-art result on one sub-dataset and very competitive results on the others. In addition, it also shows promising results in the consecutive RUL prediction following the degradation process of components.**

I. Introduction

As industrial equipment becomes more and more complex, more sensors are needed to monitor the state of the equipment. With the increase of monitoring data availability, a number of approaches that exploit this has been proposed to support decisions in industrial applications such as scheduling and maintenance management [1]. In manufacturing industries, the achievement of optimal maintenance planning can decrease the costs that are due to inefficient maintenance strategies such as preventive maintenance (PvM). PvM is to perform maintenance actions according to a predefined schedule determined by the age of components. Although PvM is still a common strategy to avoid a great deal of downtime from unexpected failures, this strategy requires excessive maintenance in order to achieve prevention [2]. On the other hand, to manage maintenance, predictive maintenance (PdM) utilizes the monitoring data that are collected by sensors as well as the current age of the target components. The goal of this monitoring is to determine the condition of the components, and this can be achieved by observing degradation-based measures [2]. The condition of the components can be predicted by analysing the degradation patterns in the collected sensory information.

Machine learning (ML) has been widely used to analyse such patterns in the data for PdM [3]. ML techniques can learn the complicated relation between the analysed historical pattern and the condition of the components, so that they can predict the time of future failure based on the monitoring data from sensors. This can also be considered as remaining useful life (RUL) estimation of the industrial equipment, since the RUL is defined as the time left until the predicted failure. In "Industry 4.0", the prognostics and health management (PHM) of industrial components is considered as a key concept for PdM [3], [4]. As ML developed, the above RUL prediction has become a mainstream element of the PHM in the context of PdM research.

The RUL prediction can be interpreted as the regression problem regarding components' condition over time. Recently, as an alternative to traditional ML, many deep learning (DL)-based approaches have been proposed to solve this problem because they can offer an end-to-end learning framework for the prediction without the complex feature engineering which is the major challenge of traditional ML [3]. Convolutional neural network (CNN) can be used to estimate the RUL of components [5]. As an alternative approach, recurrent neural networks (RNN), including long short term memory (LSTM), are able to predict the RUL by directly recognizing temporal patterns of the data instead of extracting their convolutional features [6]. In addition, deeper networks were proposed to improve the accuracy of RUL prediction [7].

Recently, the wireless sensor networks (WSN) community has used the analysis of sensor reading predictions to detect anomalies resulting from the sensors or the environment [8], [9]. These methods take into account that the sensor signals reflect the state of the environment and the future state can be predicted based on this. Therefore, the deviation between the prediction and the following measurement, the prediction error, can represent the degree of the unexpected behavior of the environment. In our work, similarly, we consider that the sensor data indicate the state of the physical properties which are monitored by the sensors. The degree of unexpected value change in each property is then closely related to the components' condition, which is related to the RUL.

This paper introduces a way of employing the multi-head CNN-LSTM network to predict the RUL of components based on multivariate time series collected by monitoring sensors. Furthermore, we improve the accuracy of the RUL prediction by combining the network with prediction error analysis (PEA). A recursive least squares (RLS) and a single exponential smoothing (SES) were used to yield the prediction

ISSN 2305-7254

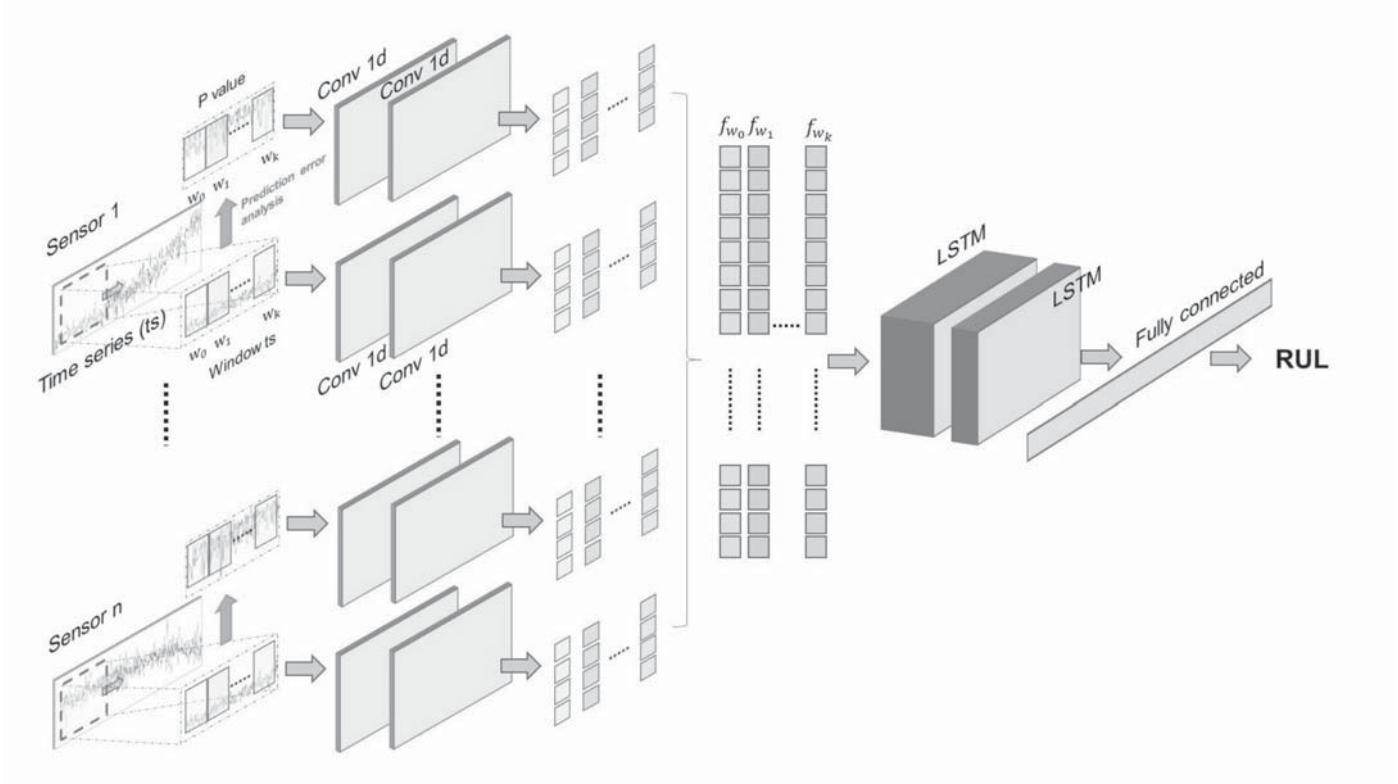


Fig. 1. Framework of multi-head CNN-LSTM with prediction error analysis

errors for every time cycle in each time series. We then use the combination of the output of the above methods as an additional information for the RUL prediction of the multi-head CNN-LSTM network.

The rest of paper is structured as follows: Section II describes the used network architecture and its combination with PEA. In Section III, the experimental setup and the results of the experiments are discussed. Finally, our conclusions are outlined in Section IV.

II. Method

A. Multi-head CNN-LSTM

In recent years, the combination of CNN and RNN has shown promising results in various research fields such as script identification [10], speech emotion recognition [11] and human activity recognition [12]. Based on those successes, in this paper, it is used for the RUL prediction. We employ a combination of CNN and LSTM which is inspired by the multi-head CNN-RNN originally proposed in [13] for multivariate time series anomaly detection on a real industrial scenario. This framework of the network is illustrated in Fig.1. The multi-head CNN-LSTM consists of three parts: multi-convolutional heads, stacked LSTM layers, and a fully connected layer. Each sensor's data are segmented by a fixed-size window and fed into each convolutional head independently. The data are then processed by one-dimensional (1D) convolutional layers to extract features. To extract an independent feature for each sensor data, each of the convolutional layers has its own kernel for individual convolution, without sharing parameters with other branches. The stacked LSTM layers recognize the temporal dependencies between the concatenated features. Lastly, the fully connected layer yields the value of predicted RUL from the output of the LSTM. The network is trained based on the mean squared error between the predicted RUL and the target RUL.

There are two major advantages for this parallel branched architecture. At first, each branch has its own convolutional layers and takes one particular time series from a certain sensor as its input, so that the learning of the convolutional filters in it specializes to one particular sensor readings. This allows each CNN to have a flexible architecture and extract a proper features that can adapt to the particular sensor. Secondly, we can exploit the additional time series when it is available by simply adding the input branch.

B. Prediction Error Analysis (PEA)

In this paper, we assume that the time series is collected by sensors, and it is used as the input for the RUL predictor. Please note that both sensor readings and the RUL predictions are in units of time (e.g., minutes or cycles). This means that we assume that all the sensors update their reading at the end of each time cycle, so that the sensed measurements of all the sensors are aligned in time. Because we consider data-driven methods that merely use historical data to predict RUL, the accuracy of the prediction is determined by the analysis of those sensor data and the pattern recognition performance of the deep learning architecture. Since our deep learning-based approach does not require any external analysis, such as feature engineering, we cannot ensure that the feature extraction occurring within the CNN always fully exploits the information in the input time series. Moreover, it is a very challenging task to find such a hyper-parameter setting of the

CNN, because the final output of the network is determined by the parameter specification of the LSTM as well as CNN. Hence, instead of entirely relying on the above network, we incorporate further analysis of the time series in terms of prediction errors into the network.

The procedure for obtaining the prediction error is described as follows: we consider a sequence of data $\mathbf{X}$ that are determined by a moving window over the sensor readings, i.e., $\mathbf{X} = (x_0, x_1, \ldots, x_d)$, where $d+1$ is the size of the window. Then, we set the last measurement x_d in this sequence as the target value of our prediction model. The remaining data, $\mathbf{X} \backslash x_d = (x_0, x_1, \ldots, x_{d-1})$, are used as the input of the model in order to predict the last measurement $\hat{x}_d$. Finally, the prediction error of the data sequence is defined as the difference between the measurement and the output of the model, i.e., $\epsilon = x_d - \hat{x}_d$. To convert the sensor data into prediction errors, this procedure is repeated until the moving window covers the entire time series.

We use two time series prediction methods, namely recursive least squares (RLS) and a single exponential smoothing (SES):

1) RLS: RLS is an approach that recursively attempts to estimate the coefficients that minimize the least square error between the response of a linear model and the target value. We can describe the linear model for the input sequence as:

$$\hat{x}_d = \boldsymbol{\beta} \cdot (\mathbf{X} \backslash x_d) \tag{1}$$

where the weights $\boldsymbol{\beta}$ are $(\beta_0, \ldots, \beta_{d-1})$. For the estimation of the weights, we adapt the recursive iteration algorithm used in [8], where RLS is used as a prediction model instead of linear least squares estimation (LLSE) because of computational resource limitations in WSN environments. Even though we did not assume this limitation in our work, we employ the RLS also for saving computational cost. In fact, the RLS enables to avoid adding an extra computational cost to the heavy computations of the multi-head CNN-LSTM network.

Algorithm 1 illustrates how the weights and the prediction error are updated at every sample step t, i.e., every time cycle in our case. At first, the algorithm is initialized by determining the value of the forgetting factor α and a scalar factor δ, that are used for initializing the inverse auto-correlation matrix P. Additionally, we determine $\boldsymbol{\beta}_{0,i}$ as the initial weights, where i indicates the index of the vector elements. At each iteration, the prediction error ϵ is defined as the difference between the observation $\hat{x}$ and the output of the model at the previous iteration with the current inputs. Then, to estimate the weights $\boldsymbol{\beta}$ and the inverse auto-correlation matrix P, the algorithm takes the linear correlation of the input signals $\mathbf{X}$, which is denoted by K. By using it, the update gain μ and the learning rate δ can then be computed. After that, the scaling factor for $\boldsymbol{\beta}$ update, κ, is determined by multiplying K by δ. Finally, $\boldsymbol{\beta}$ and P are updated based on the above temporary parameters, so that these are used for the computations of the following iteration. For further details, we follow the specification described in [14].

Algorithm 1 Recursive least squares [9]

Init $\delta > 1$, $\alpha > 1$, $\boldsymbol{\beta}_{0,i} = \{0\}$, $P_0 = \delta I$
for each sample step t **do**
  $\epsilon_t \leftarrow \hat{x}_t - \mathbf{X}_t^T \boldsymbol{\beta}_{t-1}$
  $K_t \leftarrow P_{t-1} \mathbf{X}_t$
  $\mu_t \leftarrow \mathbf{X}_t^T K_t$
  $\delta_t \leftarrow \frac{1}{\alpha + \mu_t}$
  $\kappa_t \leftarrow \delta_t K_t$
  $\boldsymbol{\beta}_t \leftarrow \boldsymbol{\beta}_{t-1} + \kappa_t \epsilon_t$
  $P_t \leftarrow \frac{1}{\alpha}\left[P_{t-1} - \delta_t K_t K_t^T\right]$
end for

2) SES: Exponential smoothing is a technique for smoothing univariate time series with exponential functions [15]. It can be used for predicting time series, and the prediction is defined as a weighted average of the last data point of the input sequence and the previous prediction:

$$\hat{x}_d = \alpha x_{d-1} + (1-\alpha)\hat{x}_{d-1} = \hat{x}_{d-1} + \alpha(x_{d-1} - \hat{x}_{d-1}) \tag{2}$$

where α is the smoothing factor ranging between 0 and 1. We can expand (2) to represent the exponential weights form as:

$$\hat{x}_d = \alpha[x_{d-1} + (1-\alpha)x_{d-2} + \cdots + (1-\alpha)^{d-2} x_1] + (1-\alpha)^{d-1} x_0. \tag{3}$$

The most recent data have the highest weight, and the exponentially decreasing weights are assigned to the older data in order. We initialize the smoothing factor α using a combination of grid search and the first value of the data, and adapt the model by estimating it in the direction of maximizing the log-likelihood.

We use the SES rather than the triple exponential smoothing, also called Holt-Winters exponential smoothing (HWES)[15], for the following reason: we assume that the length of the data sequence $d+1$ is small enough so that its trend (or seasonal pattern) cannot be reliably recognized, i.e. the sensor readings do not have a clear trend or seasonality in a certain period of time.

C. Multi-head CNN-LSTM with PEA

Once the prediction errors are derived by using the RLS and the SES respectively, we convert the prediction error of each time cycle into a p-value, in order to use it as the input for the multi-head CNN-LSTM network. The procedure can be summarized as follows: the two statistics of the prediction errors, mean μ and standard deviation σ, are established by using the exponentially weighted moving average (EWMA) [16]. These statistics are then used to calculate the p-value with a predefined confidence level 95%, similar to [9]. Finally, we combine the two p-values derived from RLS and SES via the Fisher's method:

$$X_{2k}^2 \sim -2 \sum_{i=1}^{k} \ln(p_i)$$

where k is the number of prediction models used, which is two in our work, and p_i refers to the p-value produced by the i-th prediction model. The method implicitly assumes that the models are independent. The p-values of those independent models are combined into one test statistic X^2, which has a chi-squared distribution with $2k$ degrees of freedom. In turn, the Fisher's method provides another p-value based on the above procedure, and this is the final output of the PEA.

Taking the advantages of the multi-head CNN-LSTM described in Section II-A, we can use the output of the time series PEA as an additional input for the network. As shown in Fig.1, we add one convolutional branch per each head, and then we feed the p-values produced by PEA into the added branches as the additional inputs. In this way, we can combine our further analysis on the sensor data, i.e., the degree of unusual dynamics in sensor readings, into the network for improving the accuracy of RUL prediction.

III. Evaluation

In this section, we describe the evaluation setup and procedure that are used to compare the performance of our method with the existing methods, on the C-MAPSS dataset. We first outline the description of the dataset. Then, we introduce two evaluation metrics which help compare the prediction performance. In addition, the parameters of our network are specified, and the details of training, validation and test process are described. Finally, we report experimental results and their analysis.

A. Benchmark dataset

In our experiment, the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS)[17] is used to evaluate our proposed method. The dataset consists of various NASA turbofan engine degradation simulations under four different simulation settings, included in four sub-datasets: FD001, FD002, FD003 and FD004, respectively. In these sub-datasets, each engine's data include a multivariate time series collected from 21 sensors. Among them, we used only 14 time series that show trend over cycles, as it was done in [18], [19]. While run-to-failure simulations were assumed for collecting sensor readings in the training sets, the test simulations were terminated before the failure, so that the RUL of each test engine was required to be predicted in the last cycle. An overview of four sub-datasets is described in Table I. All engines in the dataset are assumed to start their operation from a healthy condition, and keep operating afterwards. However, for most engines, the degradation does not start at the beginning of the time series, and the RUL does not decrease across all cycles. Therefore, we consider the piece-wise linear function for the degradation profile of the RUL, based on [20]. Regardless of the maximum cycles shown in Table I, we set all the target RUL greater than 125 to 125, that is the fixed value smaller than the minimum number of cycles in the training set.

B. Evaluation metrics

The used multi-head CNN-LSTM network was trained in order to accurately predict the RUL of the test samples. Therefore, the performance of our method is determined by the error between the predicted RUL and the target RUL. In order to compare the performance with other methods in literature, we considered two following metrics:

TABLE I. C-MAPSS dataset overview.

Sub-dataset	FD001	FD002	FD003	FD004
Number of engines in training set	100	260	100	249
Number of engines in test set	100	259	100	248
Max/min cycles in training set	362/128	378/128	525/145	543/128
Max/min cycles in test set	303/31	367/21	475/38	486/19
Operating conditions	1	6	1	6
Fault modes	1	1	2	2

1) RMSE: The root-mean-square error (RMSE) is a way to measure the errors. It represents the square root of the mean of the squared errors, calculated as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(RUL_i^{predicted} - RUL_i^{target})^2} \quad (4)$$

where N is the total number of test samples, which are fed into our network during the test.

2) Score: The second metric is calculated as:

$$Score = \sum_{i=1}^{N} SF_i, \quad SF = \begin{cases} e^{-\frac{d_i}{13}} - 1, & d_i < 0 \\ e^{\frac{d_i}{10}} - 1, & d_i \geq 0 \end{cases} \quad (5)$$

where d_i represents an error, $RUL_i^{predicted} - RUL_i^{target}$. And, N is the total number of test samples. This value corresponds to the error on the asymmetric score function, assigned to the score of each prediction. The score is then summed up over all the test samples. This scoring function, discussed in [21], was proposed to take into account different penalties between late and early predictions, because an early prediction is usually better than a late one in terms of preventing failures.

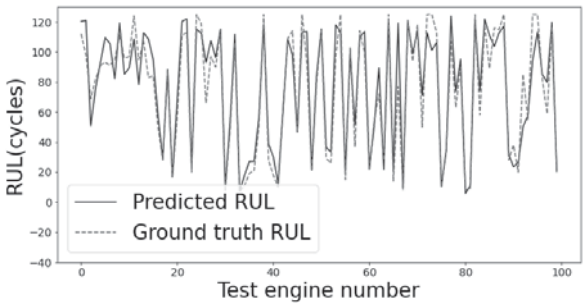
C. Deep network architecture and training details

This work employs a series combination of CNN and LSTM networks for the RUL prediction. The performance of the networks is affected by the parameter specification of both the convolutional and recurrent layers. For each convolutional head, the input data are fixed-length univariate time series. We extract these input data from the entire time series by applying a moving window. The size of the window depends on the dataset, and it cannot be larger than the minimum number of cycles in the test set, which is shown in Table I. Thus, we set the length of the time series windows (W_{ts}) as 30, 21, 38 and 19 for FD001, FD002, FD003 and FD004 respectively. Before feeding the extracted time series into the network, we divide it into fixed-length segments with a segmenting window. The length of the segmenting window (W_{sg}) is set to 3 for all the input branches. The number of input segments (N_{sg}) for each branch can be determined by:

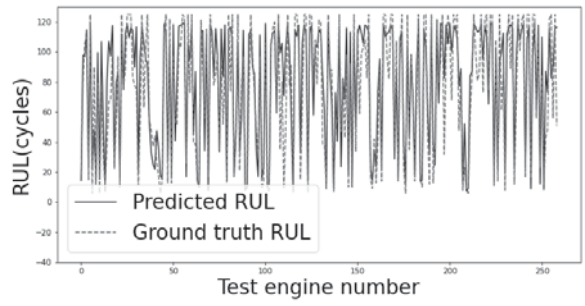
$$N_{sg} = \frac{W_{ts} - W_{sg}}{stride} + 1 \quad (6)$$

where the $stride$ is set to 1.

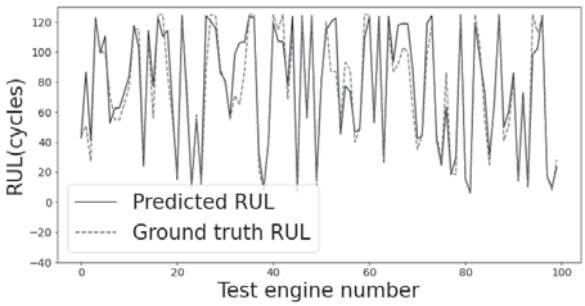
In the convolutional layer, 1D convolution is applied to every segment of the input. We chose the kernel size of 1D convolution as 3, with zero padding. Each convolutional branch consists of two stacked 1D convolutional layers with two filters for each layer. In the above branches, the convolutional layers and filters are independent, which means that each filter of the branches has the same shape but does not share its parameters with the other branches. Therefore, the number of parameters to be trained in multi-head CNN networks has to be multiplied by the number of sensors. The two stacked LSTM layers are responsible for recognizing the temporal patterns throughout the concatenated features extracted from each segment. Considering the training time, we used different network specifications for different sub-datasets. The first



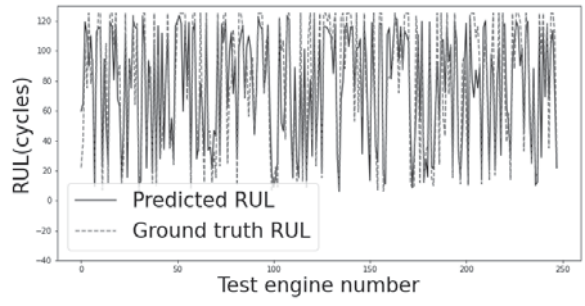
(a)



(b)



(c)



(d)

Fig. 2. Predicted RUL of each engine in the last given cycle by multi-head CNN-LSTM with PEA: (a) FD001 dataset, (b) FD002 dataset, (c) FD003 dataset, (d) FD004 dataset

LSTM layer contains $10 \cdot N_{sg}$ units, and the following LSTM layer has $5 \cdot N_{sg}$ units.

The architecture of the proposed method, the multi-head CNN-LSTM with PEA, has a duplicated additional convolutional branch for each sensor. As discussed earlier, the extracted time series before segmenting is converted into a

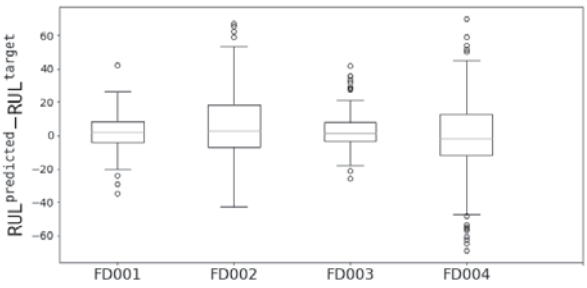


Fig. 3. Box plot of the RUL prediction error, defined by $RUL^{predicted} - RUL^{target}$, for the test sets

p-value with two different prediction models, RLS and SES. We set the input size of both models as 3, so that we derive the deviation between the output and the observation at the next time cycle. The model is then adapted with the prediction error for the next prediction. Therefore, the model does not provide the prediction errors of the first three time cycles in the time series window. The p-values for those samples are assigned a zero value. To derive the p-value from the prediction error, we used the EWMA. The mean μ and standard deviation σ of the prediction errors are initialized on the data from the same sensor measuring different engines. We combined the p-values derived from different prediction models via the Fisher's method. This method yields another p-value for each time cycle, and it is fed into the network after segmenting.

All the networks used in our experiments are trained on all the C-MAPSS sub-datasets, and the piece-wise linear function with maximum value 125 is used for the target RUL. We did not perform any pre-processing of the time series but we normalized its values in order to use them as the network inputs. As our architecture is designed for a regression task, the mean squared error is used as a loss function. The RMSprop algorithm with the leaning rate 0.001 is used as optimizer. In addition, the batch size is set to 500, and the number of training epochs is limited to 35.

To avoid overfitting, we use early stopping with 10-fold cross validation: at first, we prepare 10 identical networks; then, each of them is trained on the different subsets of the entire training set. Each model has a different validation set and stops its training when there is no improvement in the validation loss for the last 5 epochs. Finally, we take the best model that has the smallest validation loss compared with the others. In [19], the authors trained their network until it reached the maximum number of training epochs without validation, so that they used the entire training set for training. On the other hand, our network is trained with only 90 % of the training samples, while the remaining 10 % are used for validation. It should be noted that although we use a smaller number of samples for training, our model can offer a stable performance on the test set because validation helps training generalize to previously unseen data.

D. Experimental results

In the experiments, the original split of the test set in the C-MAPSS dataset is used to evaluate the performance of the tested methods on previously unseen data. We tested the multi-head CNN-LSTM without and with PEA (the latter being our

TABLE II. RUL PREDICTION COMPARISON WITH OTHER METHODS, BASED ON RMSE.

Methods	Year	RMSE				
		FD001	FD002	FD003	FD004	Sum
MLP [5]	2016	37.36	80.03	37.39	77.37	232.15
SVR [5]	2016	20.96	42.00	21.05	45.35	129.36
CNN [5]	2016	18.45	30.29	19.82	29.16	97.72
LSTM [6]	2017	16.14	24.49	16.18	28.17	84.98
MODBNE [22]	2017	15.04	25.05	12.51	28.66	81.26
RNN [7]	2018	13.44	24.03	13.36	24.02	74.85
DCNN [7]	2018	12.61	22.36	12.64	23.31	70.92
BiLSTM [23]	2018	13.65	23.18	13.74	24.86	75.43
Semi-supervised DL [24]	2019	12.56	22.73	**12.10**	22.66	70.05
DAG network [19]	2019	**11.96**	20.34	12.46	**22.43**	**67.09**
Multi-head CNN-LSTM [13]	2020	13.27	**19.49**	13.21	23.89	69.86
Proposed method	2020	12.19	19.93	12.85	22.89	67.86

TABLE III. RUL PREDICTION COMPARISON WITH OTHER METHODS, BASED ON SCORE.

Methods	Score			
	FD001	FD002	FD003	FD004
MLP [5]	$1.80E4$	$7.80E6$	$1.74E4$	$5.62E6$
SVR [5]	$1.38E3$	$5.90E5$	$1.60E3$	$3.71E5$
CNN [5]	$1.29E3$	$1.36E4$	$1.60E3$	$7.89E3$
LSTM [6]	$3.38E2$	$4.45E3$	$8.52E2$	$5.55E3$
MODBNE [22]	$3.34E2$	$5.59E3$	$4.22E2$	$6.56E3$
RNN [7]	$3.39E2$	$1.43E4$	$3.47E2$	$1.43E4$
DCNN [7]	$2.74E2$	$1.04E4$	$2.84E2$	$1.25E4$
BiLSTM [23]	$2.95E2$	$4.13E3$	$3.17E2$	$5.43E3$
Semi-supervised DL [24]	$2.31E2$	$3.37E3$	$\mathbf{2.51E2}$	$\mathbf{2.84E3}$
DAG network [19]	$\mathbf{2.29E2}$	$\mathbf{2.73E3}$	$5.53E2$	$3.37E3$
Multi-head CNN-LSTM [13]	$3.30E2$	$2.88E3$	$4.01E2$	$6.52E3$
Proposed method	$2.59E2$	$4.35E3$	$3.43E2$	$4.34E3$

proposed method). For each engine in the test set, we predicted the RUL of the engine at the last time cycle.

The test results for the four C-MAPSS sub-datasets are depicted in Fig. 2, where the horizontal axis shows the number of test engines, which is listed in Table I, while the vertical axis indicates the RUL. In each subfigure, the deviation between two graphs represents the error of our RUL prediction. The figure shows that a large number of predictions have a small error (less than 20 time cycles), but there are only few outliers that have an error larger than 40 time cycles.

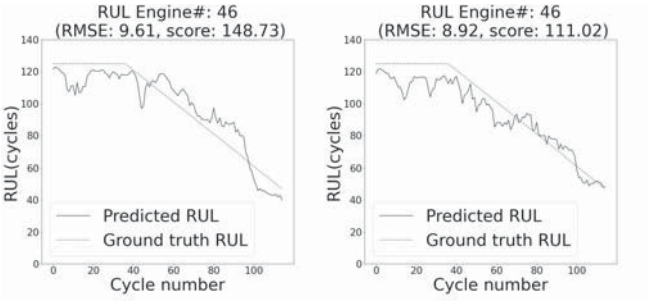
Fig. 3 illustrates the distribution of the error for each test set. The box plot for FD002 and FD004 larger min-max range and inter-quartile range (IQR) than for the other two test sets. This outlines that it is more challenging to make accurate predictions on FD002 and FD004 than on FD001 and FD003, because the former two are simulated under six different operating conditions, while the latter two are simulated in only one condition. Fig. 3 also shows that the test for FD003 and FD004 results in more outliers (i.e., samples with large prediction errors) with than the test for FD001 and FD002. This indicates that FD003 and FD004 are more challenging in terms of generalization, since the outliers are quite likely test samples that were not covered in the training set.

The RUL prediction error, described in Fig. 2 and Fig. 3, is used for calculating the RMSE and score in order to measure the performance of our method. The value of those two metrics is reported and compared to other methods in Table II and Table III. As shown in Table II, the methods introduced over the past four years managed to provide gradually lower RMSE. For all four sub-datasets, the deep learning-based methods give significantly better results in terms of RMSE w.r.t. those that do not make use of deep neural networks. In Tables II and III, the result of the proposed method is competitive compared to the recent state-of-the-art methods from the literature [19], [24]. Although the multi-head CNN-LSTM [13] provides the lowest RMSE when it is used for predicting the RUL on FD002, this conventional use of multi-head CNN-LSTM does not achieve state-of-the-art performance on the other sub-datasets.

The last row of both Tables, which shows the results of the proposed method, highlights instead that using PEA improves in general the performance of the RUL prediction model based on multi-head CNN-LSTM without PEA [13]. By using the p-values of PEA as additional inputs to the multi-head CNN-LSTM network, it is possible to decrease RMSE of around 1 and produce better scores for FD001 and FD004, while the degree of improvement is slightly reduced for FD003.

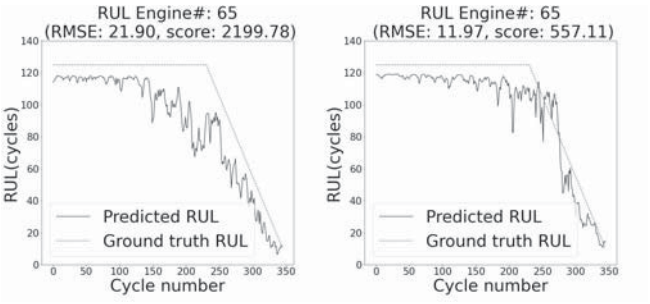
Overall, when we consider the sum of RMSE on the four sub-datasets in the last column of Table II, our proposed method outperforms all other methods except the one using the DAG network [19]. There are two aspects that lead our method to outperform the compared algorithms. Firstly, the use of a multi-head CNN-LSTM [13] for RUL prediction already provides lower RMSE w.r.t. that obtained using either CNN [5], [7] or RNN [7], [23] alone. This result is worth to check because it shows that a serial combination of CNN and LSTM, which are capable of extracting convolution and temporal features, respectively, is a good approach to predict RUL from multivariate time series data. However, the DL-based data-driven methods discussed above have some limitations: in particular, their performance are dependent on the amount and quality of labeled training data and the pattern recognition capability of the DL architecture. In this context, our solution, using external analysis of unexpected changes in sensor readings via PEA, compensates for the above limitations. Although the degree of unpredicted changes in the time series of sensed measurements is a crucial feature for RUL prediction, it is very hard to be extracted and learned by the DL-based data-driven methods. This is because it requires either an extremely large number of labeled data, or unnecessarily deeper networks to recognize such a pattern of unpredicted behavior. Instead of entirely relying on the DL network, we extract the above feature based on the PEA methods introduced in Section II-B. This feature can then be reflected i nto t he R UL predictor without increasing the depth of the network or needing a larger number of labeled training samples, and allowing to achieve an overall improved performance.

Despite these clear advantages, it should still be noted that in the case of FD002 the proposed method gives slightly higher RMSE and increases the score compared to the multi-head CNN-LSTM without PEA [13]. This indicates that in this case the LSTM layers are not able to recognize a meaningful temporal pattern in the features of the p-values extracted by the CNN layers. The reason is that a trend of p-values over time cycles may not have been detected sufficiently b y the combination of RLS and SES. In the proposed method, we assumed that an unexpected changes in the sensor data increases over time cycle after the engine degradation starts. However, each simulated time series in FD002 is likely to follow the trend monotonously without any unexpected behavior, even as it gradually approaches the engine failure. Thus, while this monotonous time series is easier to analyze for the network and allows to achieve the lowest RMSE, it does not provide enough



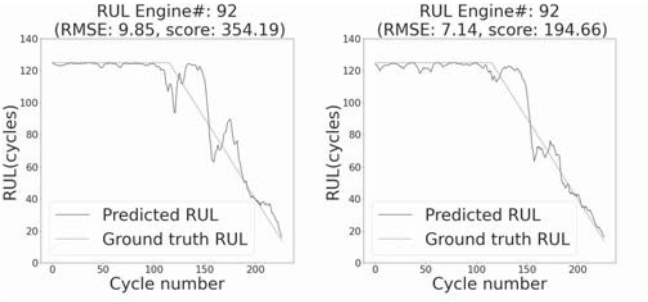
(a) (b)

Fig. 4. The degradation profile of predicted RUL of engine #46 in FD001 dataset: (a) multi-head CNN-LSTM, (b) multi-head CNN-LSTM with PEA



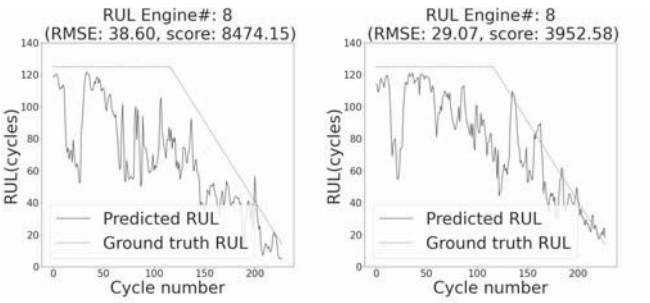
(a) (b)

Fig. 5. The degradation profile of predicted RUL of engine #65 in FD002 dataset: (a) multi-head CNN-LSTM, (b) multi-head CNN-LSTM with PEA



(a) (b)

Fig. 6. The degradation profile of predicted RUL of engine #92 in FD003 dataset: (a) multi-head CNN-LSTM, (b) multi-head CNN-LSTM with PEA



(a) (b)

Fig. 7. The degradation profile of predicted RUL of engine #8 in FD004 dataset: (a) multi-head CNN-LSTM, (b) multi-head CNN-LSTM with PEA

prediction error information to improve the RUL prediction.

Finally, in addition to predict the final RUL of each engine, we tested the proposed method on the entire time series of each test engine. The purpose of this experiment is to evaluate the performance of our method when used for estimating engine degradation. As shown in Fig. 4-7, the output of this test can be visualized by the graph of the engine degradation over cycles. Because our method uses a fixed-length time series as an input for predicting the RUL at the last time cycle, the data for the first W_{ts} cycles of each engine are not used as test sample. Therefore, the cycle number of each sub-figure starts from W_{ts}. Each figure consists of two sub-figures (a) and (b), that present the result of the test of multi-head CNN-LSTM without PEA [13] and with PEA (i.e., the proposed method), respectively. For each test, we illustrate the trajectory of both the predicted RUL and the ground truth RUL, and the performance is measured by the RMSE and the score over all the cycles of each engine. To compare the performance of the two methods at different engine conditions, we selected an engine that has a clear degradation in its data. As we can note by comparing Fig. 4-7(b) to Fig. 4-7(a) respectively, the proposed method provides lower RMSE and score than those of multi-head CNN-LSTM without PEA [13]. In particular, it should be noted that before the actual degradation of the RUL starts, the benefit of using the PEA is trivial. The time series prediction error is in fact very small in this period, because the time series is rather stable within a small range of values without trend. However, after the RUL curve starts to decrease linearly over cycles, the PEA substantially improves the prediction accuracy compared to the beginning of the time series.

IV. Conclusion

In this paper, we proposed a novel method improving the RUL prediction accuracy by combining multi-head CNN-LSTM with time series PEA. The RLS and SES were used to extract the prediction error over time from multiple time series, and the two prediction error's p-values were then combined into another p-value by applying the Fisher's method. Finally, we used the output of the above PEA as an additional input for the multi-head CNN-LSTM, so that the degree of unexpected change in time series can be used for the RUL prediction.

To evaluate the above methods, we performed experiments on the widely used public C-MAPSS dataset. Our proposed method achieved state-of-the-art results on the FD002 sub-dataset. While the multi-head CNN-LSTM gave slightly inferior performance on the other sub-datasets compared to other recent methods, the combination of the network with PEA showed a competitive performance for all the sub-datasets. These experiments confirmed that using PEA significantly improves the RUL prediction accuracy compared to not using it.

In addition, we tested our trained model of the proposed method on the entire time series of each test engine. The test results of the selected engines showed that the PEA allows the network to predict more accurately the degradation process of the engine. Furthermore, the predicted RUL graph over time cycles indicated that it is more advantageous to use the proposed method when the engine condition is somewhat close to the failure than to use it at the early stage of the engine life.

ACKNOWLEDGEMENT

This work was co-funded by BlueTensor S.r.l and Trentino Sviluppo

REFERENCES

[1] G. A. Susto, A. Schirru, S. Pampuri, S. McLoone and A. Beghi, "Machine Learning for Predictive Maintenance: A Multiple Classifier Approach", *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 812-820, 2015.

[2] K. A. Kaiser and N. Z. Gebraeel, "Predictive Maintenance Management Using Sensor-Based Degradation Models", *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 39, no. 4, pp. 840-849, 2009.

[3] W. Zhang, D. Yang and H. Wang, "Data-Driven Methods for Predictive Maintenance of Industrial Equipment: A Survey", *IEEE Systems Journal*, vol. 13, no. 3, pp. 2213-2227, 2019.

[4] D. Wang, K. Tsui and Q. Miao, "Prognostics and Health Management: A Review of Vibration Based Bearing and Gear Health Indicators", *in IEEE Access*, vol.6, pp. 665-676, 2018.

[5] G. S. Babu, P. Zhao and X.-L. Li, "Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life", *Database Systems for Advanced Applications*, pp. 214-228, 2016.

[6] S. Zheng, K. Ristovski, A. Farahat and C. Gupta, "Long Short-Term Memory Network for Remaining Useful Life estimation", *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 88-95, 2017.

[7] X. Li, Q. Ding and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks", *Reliability Engineering & System Safety*, vol.172, pp. 1-11, 2018.

[8] H. H. W. J. Bosman, A. Liotta, G. Iacca and H. J. Wörtche, "Anomaly Detection in Sensor Systems Using Lightweight Machine Learning", *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 7-13, 2013.

[9] H. H. W. J. Bosman, G. Iacca, A. Tejada, H. J. Wörtche and A. Liotta, "Ensembles of incremental learners to detect anomalies in ad hoc sensor networks", *Ad Hoc Networks*, vol.35, pp. 14-36, 2015.

[10] A. K. Bhunia, A. Konwer, A. K. Bhunia, A. Bhowmick, P. P. Roy and U. Pal, "Script identification in natural scene image and video frames using an attention based Convolutional-LSTM network", *Pattern Recognition*, vol.85, pp. 172-184, 2019.

[11] J. Zhao, X. Mao and L. Chen, "Speech emotion recognition using deep 1D & 2D CNN LSTM networks", *Biomedical Signal Processing and Control*, vol.47, pp. 312-323, 2019.

[12] K. Xia, J. Huang and H. Wang, "LSTM-CNN Architecture for Human Activity Recognition", *IEEE Access*, vol.8, pp. 56855-56866, 2020.

[13] M. Canizo, I. Triguero, A. Conde and E. Onieva, "Multi-head CNN–RNN for multi-time series anomaly detection: An industrial case study", *Neurocomputing*, vol.363, pp. 246-260, 2019.

[14] G. E. Bottomley and S. T. Alexander, "A novel approach for stabilizing recursive least squares filters", *IEEE Transactions on Signal Processing*, vol.39, pp. 1770-1779, 1991.

[15] C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages", *International Journal of Forecasting*, vol.20, pp. 5-10, 2004.

[16] S. W. Roberts, "Control Chart Tests Based on Geometric Moving Averages", *Technometrics*, 1:3, pp. 239-250, 1959.

[17] A. Saxena, K. Goebel, D. Simon and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation", *2008 International Conference on Prognostics and Health Management*, pp. 1-9, 2008.

[18] C. Zheng, W. Liu, B. Chen, D. Gao, Y. Cheng, Y. Yang, X. Zhang, S. Li, Z. Huang and J. Peng, "A Data-driven Approach for Remaining Useful Life Prediction of Aircraft Engines", *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 184-189, 2018.

[19] J. Li, X. Li and D. He, "A Directed Acyclic Graph Network Combined With CNN and LSTM for Remaining Useful Life Prediction", *IEEE Access*, vol.7, pp. 75464-75475, 2019.

[20] F. O. Heimes, "Recurrent neural networks for remaining useful life estimation", *2008 International Conference on Prognostics and Health Management*, pp. 1-6, 2008.

[21] A. Saxena, K. Goebel, D. Simon and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation", *2008 International Conference on Prognostics and Health Management*, pp. 1-9, 2008.

[22] 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*, vol.28, pp. 2306-2318, 2017.

[23] J. Wang, G. Wen, S. Yang and Y. Liu, "Remaining Useful Life Estimation in Prognostics Using Deep Bidirectional LSTM Neural Network", *2018 Prognostics and System Health Management Conference (PHM-Chongqing*, pp. 1037-1042, 2018.

[24] A. Listou Ellefsen, E. Bjørlykhaug, V. Æsøy, S. Ushakov and H. Zhang, "Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture", *Reliability Engineering & System Safety*, vol.183, pp. 240-251, 2019.