Novel approach to estimate Remaining Useful Life in Condition Based Maintenance

Samira Abderrezek
ReLa(CS)^2 Laboratory
University of Oum El Bouaghi P.O. Box 358
04000 Oum El Bouaghi, Algeria
s.abderrezek@centre-univ-mila.dz

Abdelhabib Bourouis
ReLa(CS)^2 Laboratory
University of Oum El Bouaghi P.O. Box 358
04000 Oum El Bouaghi, Algeria
bourouis.abdelhabib@univ-oeb.dz
ORCID: 0000-0002-4592-4042

***Abstract*—Deep learning is an efficient tool for Remaining Useful Life (RUL) estimation, which is crucial for intelligent prognosis and Condition-Based Maintenance (CBM) strategies. To achieve this task, Bidirectional long short-term memories have been preferred for their ability to identify patterns of temporal sequences independently, and Convolutional Autoencoder is performed in extracting features. To benefit from the advantages of these two deep learning models, this paper proposes their hybridization. We investigate the best configuration by varying the values of the hyperparameters and evaluating their impact on the new model's performance. Finally, it is compared with other similar models in order to study the effectiveness of the approach.**

***Index Terms*—Condition-based maintenance; Remaining Useful Life (RUL), Bidirectional Long-Short Term Memory neural network, Convolutional auto-encoder neural network, C-MAPSS dataset**

I. Introduction

Intelligent prognostics and machine health management technologies have shown promising application capabilities in many industries, also called condition-based maintenance (CBM). By monitoring the installation conditions, CBM can maximize operational availability, reduce maintenance costs, and improve the reliability and security of the system. It involves the evaluation of machines' performance degradation by estimating their Remaining Useful Life (RUL). The RUL of an asset is defined as time left from the current time to the end of its useful life [1]. It is the prediction of the remaining time during which a system can perform its planned function. Generally, RUL has a significant influence on decision-making [2].

According to Lee and al in [3] RUL prediction approaches' are regrouped in Model-based approaches, data-driven approaches, and hybrid prognostic approaches. Model-based approaches require prerequisites of in-depth knowledge of the physical systems, which is not generally available in practice. Data-driven approaches, on the other hand, can model degradation characteristics based on historical sensor data. In recent years, many efficient data-based algorithms have been proposed, in particular: learning models such as neural networks, support vector machine (SVM), and hidden Markov models. Deep learning networks are emerging as a highly effective structures for pattern recognition, which can improve performance in the current intelligent prognostics. It is characterized by the deep network architecture where multiple layers are stacked in the network to fully capture the representative information from raw input data [4]. High-level abstractions of data can be modeled well with the help of complex deep structures, leading to more efficient feature extraction than shallow networks. Deep learning methods have gained great interest and achieved significant results in many fields, including image recognition [5] and speech recognition [6]. Since the raw data obtained from machinery health monitoring share similar high dimensionality with those in image processing researches, deep learning architecture has excellent potential in Prognosis Health Monitoring (PHM) and RUL estimation.

Data describing Condition-Based Maintenance (CBM) parameters are categorized as time series types. Time series prediction problems are a complex type of predictive modeling, which unlike predictive modeling of regression, also add the complexity of a dependency of sequence between input variables. A powerful type of artificial neural network (ANN) designed to handle sequence dependence is called recurrent neural networks (RNN). Bidirectional LSTM (BiLSTM) is a variant of RNN used in several recent works. It has given promising results, especially in capturing the hidden long-term temporal dependencies among time sequence signals. Another type of deep learning model is the Convolutional Neural Network (CNN) and the Autoencoder (AE). They have improved their performance in the domain. Mainly they can extract the essential local features from sequential data. Taking advantage of their characteristics, we have chosen to use them in a hybrid ANN for RUL estimation.

The rest of this paper is organized as follows: Section II presents recent related works. Section III presents the proposed model architecture, Section IV describes the dataset used for evaluating the models, the data pre-processing, the evaluation metrics, and the approach description. Finally, section V concludes this paper and discusses future work for the RUL estimation task.

This work is implemented using HPC resources of UCI-UFMC (Unit de Calcul Intensif) of University FRERES MENTOURI CONSTANTINE university and HPC of Batna2 university

II. Related work

Bidirectional LSTM, CNN, and Autoencoder networks have been the subject of several recent works. They have been used alone or in hybridization with other deep algorithms, and they have been improved their performances in the RUL estimation domain.

Recently, CNN was successfully introduced in RUL prediction for its strong capability in local feature extraction [7], [8], [9]. For example, in work presented in [10], a Deep Convolutional Neural Network (DCNN) alone is proposed for prognosis. It used the time window approach for sample preparation to achieve better feature extraction by the DCNN. The raw data collected with normalization is directly used as the proposed network inputs without any prior expertise on prognosis and signal processing, which facilitates the application of the proposed method.

Another architecture applied a CNN first and then uses LSTM for estimating RUL in [11]. A method of data augmentation is applied to improve the performance of the proposed approach. Another instance of this category combines CNN and LSTM in parallel [12]. While in a second paper [13], they integrated three fully noisy deep learning architectures, Noisy CNN and Noisy Bi-directional Gated Recurrent Unit (GRU) paths are designed in parallel, and their concatenated output is fed into the Noisy fusion center.

Authors in [14], designed a deep learning model combining CNN for extracting robust local features from the sequential input, and then Bi-directional LSTM is adopted to encode temporal information on the sequential output of CNN. In [15], authors used an LSTM Encoder-Decoder architecture in only the first Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dataset, obtaining an unsupervised health index to determine RUL.

A variational encoder-decoder is proposed in [16]. It is based on visual data analysis to predict when an in-service machine will fail. A deep Convolutional Variational AutoEncoder is used for automatically extracting performance degradation feature from multiple sensors. Furthermore, an encoder-decoder model has been proposed in [17]. The BiLSTM and CNN are used in the encoder, and fully connected networks are used to decode feature information.

A RUL prediction method for electric valves has been presented in [18]. Hybridization used a Convolutional Auto-Encoder for extracting deeper features and an LSTM for efficiency in dealing with time-series data. The authors designed a parallel structure between the outputs of the Convolutional Auto-Encoder and the original data to enrich features fed into the LSTM.

III. Description of the proposed approach architecture

The previous noticed works inspire the idea of our proposition in section II. It aims to combine the advantages of three different techniques through their integration so that the results can be aggregated to improve prediction performance. Convolutional Auto Encoder is used to extract principal significant features from the original signal and is also used for data dimension reduction. BiLSTM is used as a tool that models time series and captures bidirectional features long-term dependencies, since the time series of the study domain has a characteristic of dynamic instability and long-term dependencies. That's the reason for proposing a model which couples the convolutional autoencoder neural network (CAENN) and the Bidirectional neural network (BiLSTM NN) for aero turbofan engines in this paper. And so, the hybrid architecture has the advantage of giving high estimation accuracy and low computing burden. Our purpose in this work is to propose a hybrid model combining a set of the most successful Neural Network classes, which are Convolutional Autoencoder (CAE) and Bidirectional LSTM neural network in a hybrid solution that constitutes the most promising category of RUL estimation [19]. The first challenge is to perform the tuning of hyperparameters and the study of the effect of each one on the model performance to find the best alternative. The approach is finally compared with some algorithms in the literature and validated using the NASA C-MAPSS dataset.

Its components are:

1) **Convolutional AutoEncoder deep neural network**
   An autoencoder is a type of neural network used to learn efficient code in an unsupervised manner [20]. It consists of an encoder and decoder. The encoder learns to compress the input data into a short code, whereas the decoder learns to decompress the code into a set of data that closely matches the input data [21]. It could extract hidden features from input data and reconstruct original data with these features. It can be applied not only for dimension reduction but also for further classification, and prediction [22]. The authors, in [23], observed that the combination of Autoencoder with CNN could improve model accuracy.
   As an encoder in our model, we use two convolutional 2D layers. After each one, a maxpooling layer is used, where the central concept is the size reduction of the feature maps by selecting the maximum value for each patch of the feature map. Hence, reducing the number of model parameters and simplifying the computational complexity of the network, as well as helps in avoiding the over-fitting issue [24]. Then as a decoder, two convolutional 2D layers are used, and after each one, an upsampling layer (UpSampling2D) doubles the dimensions of the input. We have chosen $10, 10, 10, 10, 5, 2$ as filters, and $(10, 1), (10, 1), (10, 1), (10, 1), (10, 1), (1, 1)$ as kernel sizes. Zero padding is used to keep the feature map through the network and also (2,1) as kernels for each maxpooling and upsampling layers.
2) **Bidirectional LSTM deep neural network**
   BiLSTM is used in the proposed model to capture the bidirectional temporal dependencies between features. BiLSTM results in two LSTM layers instead of one on the input sequence. One access information in the

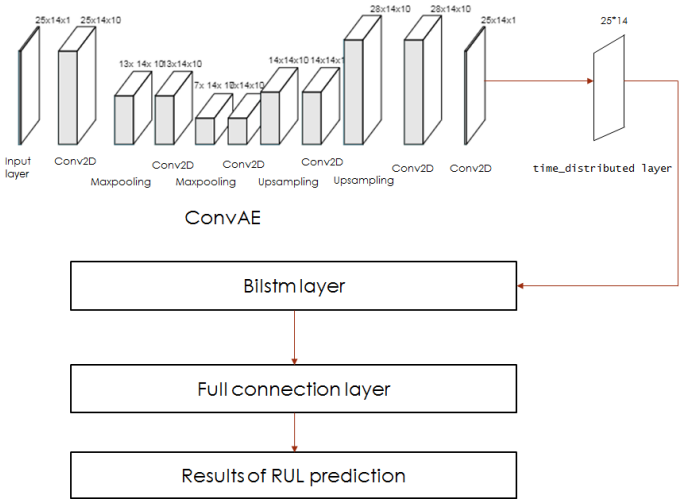

Fig. 1. CAEBi-Architecture

forward direction, and the other access it in the backward direction, which can provide additional context to the network and lead to faster and more complete learning of the problem.

Our model comprises one Bidirectional LSTM layer, followed by a dropout layer to avoid overfitting. Finally, a fully connected layer (dense layer) is applied to map the output of the Bidirectional LSTM model to the desired output size, which is equal to 1 and provided by the linear activation function.

The whole model structure is illustrated in Figure 1. The CAENN is applied firstly to perform downsampling of data to minimize its size and complexity and enhance the model generalization and learning capacity. By extracting hidden features from input data, it could easily then use them to reconstruct original data by oversampling it. Then, a distributed flatten layer is connected to the network output to hybridize with the BiLSTM network. After that, the data is introduced to the BiLSTM NN for deepening the characteristics of the information given by the CMAPSS data sources and establishing the nonlinear relationship between the multivariable's time-series and RUL. Finally, a fully connected dense layer is connected to the output of the model.

IV. Experiments and Discussion

A. Dataset description

The performance of the proposed model is evaluated using the public and widely used in the literature NASA C-MAPSS dataset [25]. It contains simulated data produced using a model-based simulation program developed by NASA.

The dataset comprises 4 subsets of multi-variate temporal data that simulate the condition monitoring data of turbofan engines under different operating conditions and failure modes. In this paper, the first sub-dataset FD001 is used. It contains 100 training engines, 100 test engines, one operating condition, one failure mod, 20630 training samples, 13 095 test samples, 21 sensors, and three operating indicators [26].

FD001 is a multiple high-dimensional consecutive time-series data. The training set records condition-monitoring data of turbofan engines from the normal condition at the beginning to complete failure at last. The condition-monitoring data of the test set ends at some point before failure occurs. The goal is to predict the number of remaining operational cycles of turbofan engines in the test set.

Only 14 sensor data are chosen here as the input data of the network, like in [10] where is discarded some sensor values because they do not vary throughout the entire engine life cycle. The picked columns are 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20 and 21.

B. Data pre-processing

1) Normalization: For an equal contribution of all features, under all conditions of operation, normalization with the min-max method is applied to a range of [-1, 1] using Equation 1 [12].

$$x_{norm}^{i,j} = \frac{2 \times x^{i,j} - x_{min}^{i,j}}{x_{max}^{j} - x_{min}^{j}} - 1, \forall i, j \tag{1}$$

Where $x^{i,j}$ denotes the i^{th} origin data point of the j^{th} sensor, and $x_{norm}^{i,j}$ norm is the normalized value of $x^{i,j}$. x_{max}^{j} and x_{min}^{j} denote respectively the maximum and minimum values of the original measurement data of the j^{th} sensor [10].

2) Time window processing: The data is processed with a sliding time window approach of size $N_f = 25$ and stride of 1. The sliding window means that the first input sample to the network takes measurements from cycles 1-25, the second 2-26, and so forth for each fleet unit. The RUL label for a sample is then simply the total number of cycles the engine can operate, minus the cycle where the window ends. [10].

Following the standard procedure adopted by other researchers, the maximum horizon of prediction for RUL (R_{early}) was limited to 125 cycles, which has an impact on the model accuracy and makes models more stable.

C. Evaluation Metrics

In order to assess the performance of the proposed prognostic model, the following functions are used.

1) **Root mean squared error (RMSE):**
RMSE is the most widely used metric for regression tasks and a performance indicator to determine the accuracy of the estimated RUL. It is given by:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} e_i^2}{n}} = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}} \tag{2}$$

Where n is the total number of samples in the test set and $e_i = \hat{y}_i - y_i$ is the error between the estimated RUL values $\hat{y}$, and the actual RUL values y for each engine within the test set. The errors are first squared before

TABLE I
AVERAGE TEST AND TRAIN LOSS RESULTS WITH DIFFERENT TIME WINDOW SIZES.

TWindow Size (WS)	Train				Test			
	MSE	MAE	RMSE	R^2	MSE	MAE	RMSE	R^2
15	360.14	15.21	16.87	0.80	263.99	11.91	14.78	0.84
20	345.89	15.02	18.10	0.76	270.94	11.65	16.16	0.81
25	239.19	12.38	13.94	0.85	190.55	10.19	13.13	0.87
30	292.21	13.43	15.12	0,82	241.46	11.50	13.35	0.86

averaging, which poses a high penalty on significant errors [27].

2) Regression score function(R^2) The scoring function is a performance measure defined by the PHM community in "The 2008 PHM data challenge competition" [26]. It is given by :

$$R^2(y, \bar{y}) = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \quad (3)$$

where $\bar{y} = \frac{1}{n} \times \sum_{i=1}^{n} y_i$ and $\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} {e_i}^2$

3) Mean squared error(MSE) Given n models of the data set, for the ith example, let p_i be the predicted value and a_i the real value. The MSE of the tested dataset is [28] :

$$MSE = \frac{\sum_{i=0}^{n} (a_i - p_i)^2}{n} \quad (4)$$

4) Mean absolute error(MAE) Robust to outliers and does not penalize the errors, S_i the predicted value of the i_th sample, and is the corresponding true value, then the mean absolute error (MAE) estimated on is defined as [29]:

$$MAE(y, \hat{y}) = \frac{1}{n_{samples}} \times \sum_{i=1}^{n_{samples}-1} |y_i - \hat{y}_i| \quad (5)$$

D. Hyper Parameter tuning

Our objective in this section is to thoroughly investigate all the hyperparameters of the model architecture that deliver the best recognition accuracy for the C-MAPSS dataset. The number of epochs here is fixed to 20.

1) Time window size parameter effect: When using deep learning models for RUL predicting, a sliding time window approach is widely employed to convert condition monitoring data stream into input samples for supervised learning. Specifically, the time window approach takes features in previous time steps as input variables, while RUL labels in the next time step as the output variable [30].

Table I shows the results obtained from the proposed CAE Bidirectional framework using values of window sizes (WS) of 10 to 30. The values of metrics function for $WS = 25$ are clearly better than the other WS values. Results obtained from larger window sizes are more efficient. A larger time window size can cover more raw information, which is the basis for further extraction features [10]. This value will be adopted in the rest of the paper.

TABLE II
EFFECTS OF LEARNING RATES.

LR	Train				Test			
	MSE	MAE	RMSE	R^2	MSE	MAE	RMSE	R^2
0.001	310.10	14.59	16.09	0.81	259.36	12.82	16.59	0.80
0.0001	249.03	12.89	14.28	0.85	218.34	10.95	13.23	0.87
0.00001	**236.89**	**12.43**	**13.87**	**0.86**	**174.34**	**10.09**	**11.59**	**0.90**

TABLE III
EFFECTS OF BATCH SIZE VALUES.

Batch Size	Train				Test			
	MSE	MAE	RMSE	R^2	MSE	MAE	RMSE	R^2
512	236.89	12.43	13.87	0.85	174.34	10.09	11.59	**0.90**
128	171.48	9.20	10.91	0.90	174.93	**9.20**	**11.29**	0.89
100	121.90	7.85	9.41	0.93	**168.59**	9.43	12.62	0.88
64	159.60	8.82	10.57	0.91	214.12	11.09	13.37	0.84

2) Effects of learning rate and batch sizes: The learning rate controls the update rate of weights in response to the estimated gradient at the end of each batch. It can significantly impact the trade-off between how quickly or how well the model learns the problem. Results in table II show that the small values of learning rate (0.00001) give better performance. Table III also shows that 128 as the value of batch size gives the best performance of the model.

3) Effects of activation function and optimizer: By using different activation functions with different optimizers, we notice that the Hyperbolic Tangent Function **Tanh** for hidden layers with "linear" activation function for the dense layer and "Adam" optimizer give the best results (Table IV).

The "Tanh" activation function has a very similar structure to the "Sigmoid" function, but it is always preferred when optimization is easier. It is frequently used in hidden layers of an ANN, as its values lie between -1 to 1. Hence, the mean of the hidden layer comes out close to 0, which helps to center data by bringing the mean close to 0. This makes learning for the next layer much more effortless and means that it will be more efficient because it has a broader range for faster learning and grading. Its inconvenience over ReLU is the problem of gradients at the ends of the function [31].

4) Effects of layers and unit numbers of BiLSTM: Table V and Table VI, show that one BiLSTM layer is enough to give best results with unit number equal to 300.

TABLE IV
EFFECTS OF ACTIVATION FUNCTION AND OPTIMIZER.

Activation Function	Optimizer	Train				Test			
		MSE	MAE	RMSE	R^2	MSE	MAE	RMSE	R^2
Tanh	**adam**	**171.48**	**9.20**	**10.91**	**0.90**	**174.93**	**9.20**	**11.29**	**0.89**
	rmsprop	160.69	8.79	10.44	**0.90**	204.53	9.78	15.02	0.82
	adadelta	141.76	7.74	9.43	0.93	245.01	10.81	17.87	0.72
ReLU	**adam**	206.85	10.43	12.18	0.88	190.22	9.91	12.44	0.88
	rmsprop	257.74	12.34	14.16	0.81	268.98	11.92	13.93	0.83
	adadelta	411.32	16.79	18.09	0.72	318.04	13.23	16.16	0.81
Sigmoid	**adam**	1750.23	37.19	38.41	/	1765.44	36.04	40.59	/
	rmsprop	1648.17	35.05	39.46	/	1750.99	37.27	38.47	/
	adadelta	1750.64	37.24	38.45	/	1651.23	35.03	39.50	/
Leaky ReLU	**adam**	127.76	8.12	9.54	0.92	**161.11**	9.38	13.19	0.86
	rmsprop	104.68	6.94	8.30	0.95	235.18	10.29	15.98	0.77
	adadelta	76.57	6.05	7.63	0.96	281.31	11.66	14.59	0.82

TABLE V
EFFECTS OF BILSTM LAYERS.

Layers	Train				Test			
	MSE	MAE	RMSE	R^2	MSE	MAE	RMSE	R^2
1	**162.02**	**8.98**	**10.64**	**0.90**	**142.90**	**8.41**	**10.36**	**0.91**
2	146.35	8.38	10.01	0.92	153.38	8.95	12.95	0.87
3	176.50	9.35	10.92	0.89	171.67	9.34	12.62	0.87
4	201.57	10.28	12.03	0.85	172.73	9.24	11.81	0.89
5	198.03	10.52	12.18	0.85	169.57	9.46	11.89	0.89

TABLE VI
EFFECTS OF BILSTM UNIT NUMBER.

Unit Number	Train				Test			
	MSE	MAE	RMSE	R^2	MSE	MAE	RMSE	R^2
128	171.48	9.20	10.91	0.88	174.93	9.20	11.29	0.89
256	162.66	9.02	10.65	0.87	182.95	9.67	11.31	0.89
300	**162.02**	**8.98**	**10.64**	0.87	**142.90**	**8.41**	**10.36**	**0.91**
350	155.87	8.71	10.41	0.89	141.30	8.55	12.13	0.89
400	191.47	9.95	11.70	0.85	192.10	9.96	11.90	0.88

5) Effects of number of layers, filters, and kernel sizes of Convolutional Auto Encoder: Results in Tables VII, VIII and IX, show that for different values of filters and kernel sizes, the values of $(10, 10)(10, 10)(10, 10)(10, 10)(10, 5)(1, 2)$ give the best performance. Furthermore, six layers of Convolutional 2D are the best.

6) Effects of validation split values: Results in Table X show that for different values of validation split, the value of 0.15 gives the best performance.

E. Final results for FD001 sub-dataset

Final results for FD001 with 20 epochs after the tuned hyperparameters are for training: MSE=166.66, MAE = 9.40, RMSE = 12.22 and R^2 = 0,85. For test: MSE=149.25, MAE=8.86, RMSE=10.07 and R^2=0.91.

Figures 2, 5 and 4 illustrated detailed comparisons between the predicted RUL values and the actual values during the whole life of engine units. It is observed that the prediction error is decreasing with usage, and units can be predicted precisely. There is more useful degradation information in the late stage than in the early stage.

Therefore, the experimental results demonstrate that the piece-wise linear degradation function is suitable for this problem. As shown in figure 3, the training RMSE and the validation RMSE decrease with the epoch in the training process. Besides, the loss value tends to be constant in the late stage. It is found that the training loss has the same trend of change as the validation loss. Therefore, there is no overfitting problem in the training process, and the selection of training hyperparameters is suitable for this problem.

TABLE VII
EFFECTS OF FILTERS.

Filters	Train			Test			
	MSE	MAE	RMSE	MSE	MAE	RMSE	R^2
10	142.71	8.31	9.91	146.35	8.92	10.15	0.91
16	155.76	8.67	10.28	171.01	9.35	12.81	0.87
32	218.97	10.59	12.38	239.39	11.45	14.34	0.84
64	132.26	8.04	9.72	182.92	9.73	13.32	0.86

TABLE VIII
EFFECTS OF KERNELS.

Kernels	Train			Test			
	MSE	MAE	RMSE	MSE	MAE	RMSE	R^2
10	142.71	8.31	9.91	146.35	8.92	10.15	0.91
6	178.64	9.55	11.16	159.88	8.94	10.27	0.90

TABLE IX
EFFECTS OF CAE LAYERS.

CAE Layers	Train			Test			
	MSE	MAE	RMSE	MSE	MAE	RMSE	R^2
7	162.02	8.98	10.64	**142.90**	**8.41**	10.36	0.91
6	**142.71**	**8.31**	**9.91**	**146.35**	**8.92**	**10.15**	**0.91**
5	173.89	9.39	11.00	162.05	8.90	10.36	0.90

TABLE X
EFFECTS OF VALIDATION SPLIT VALUES.

VS	Train				Test			
	MSE	MAE	RMSE	R^2	MSE	MAE	RMSE	R^2
0,05	159.57	8.96	10.6	0,89	169.02	9.16	12.56	0.87
0,1	**142.71**	**8.31**	**9.91**	**0,93**	**146.35**	8.92	10.15	**0.91**
0,15	**166.66**	**9.40**	**12.22**	**0,85**	149.25	**8.86**	**10.07**	**0.91**

F. Comparison of similar approaches using C-MAPSS dataset

A comparison with other similar and recent approaches presented in table XI shows that the model gives encouraging results.

V. CONCLUSION

In this paper, a model based on Convolutional Autoencoder and BiLSTM neural network for RUL estimation of C-MAPSS data is proposed. The model has the capacity of feature extraction and data reduction of CAENN and the skill of time series prediction and bidirectional temporal features extraction of BiLSTM NN.

The accuracy of the RUL prediction model exceeds 91%, while RMSE and MAE are limited to 10.07 and 8.86, respectively.

The proposed method is suitable for predicting the RUL of C-MAPSS data. Future research interests exist in optimizing different parameters of the model with a suitable meta-heuristic method. A parallel architecture is also another interesting way and maybe another network hybridization in searching for a high-efficiency model. Attention mechanism and Ensemble learning also can be applied to improve accurately predicting RUL. Furthermore, extend the study to all the other sub-datasets.

TABLE XI
RMSE COMPARISON ON C-MAPSS DATASET.

Approach	**FD001**
CNN BiLSTM [32]	10.74
LSTM [33]	16.14
BD-LSTM [34]	15.42
LSTM [34]	18.07
BD-RNN [34]	20.04
CNN [7]	18.45
MLP [34]	20.84
RVR [7]	23.79
SVR [7]	20.96
MLP [7]	37.56
DCNN [10]	12.61
LSTM [10]	13.52
RNN [10]	13.44
DNN [10]	13.56
NN [10]	14.80
[35]	12.18
Parallel LSTM CNN [12]	13.017
Noisy Parallel CNN BiGRU and FC (NPBGRU) [13]	10.44
Deep separable convolutional network (DSCN) [36]	10.95
Double channel CNN and bidirectional LSTM [37]	12.58
Gated graph neural networks (GNNs) [38]	12.14
BiLSTM MultiscaleCNN [39]	12,75
CAE with BDGRUBDLSTM [40]	9.51
CNNBDGRU [40]	11.085
The proposed method	**10.15**

TABLE XII
ABBREVIATIONS USED IN THIS MANUSCRIPT.

Adam	Adaptive Moment Estimation
ANN	Artificial Neural Network
BiLSTM	Bidirectional LSTM Neural network
CAE	Convolutional Auto Encoder
CNN	Convolutional Neural Network
C_MAPSS	Commercial Modular Aero Propulsion System Simulation
DCNN	Deep Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MLP	Multi Layer Perceptron
MSE	Mean Squared Error
DBN	Deep Belief Network
Relu	Rectified Linear Unit
RMSE	Root Mean Square Error
RMSProp	Root Mean Square Propagation
RNN	Recurrent Neural Network
R_2	R^2 correlation
RUL	Remaining Useful Life
Tanh	Hyperbolic Tangent Activation Function

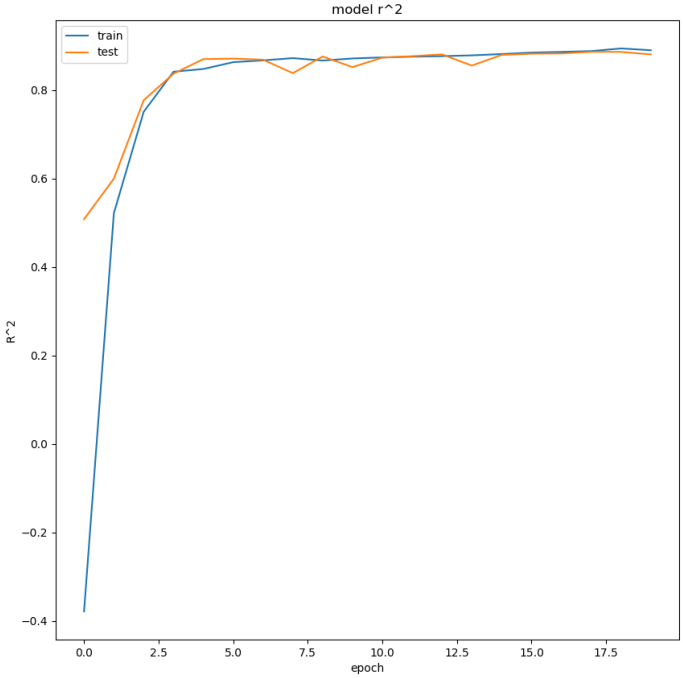

Fig. 2. R^2 values: comparison between predicted and actual RUL values for testing engines.

REFERENCES

[1] X.-S. Si, W. Wang, C.-H. Hu, and D.-H. Zhou, "Remaining useful life estimation–a review on the statistical data driven approaches," *European journal of operational research*, vol. 213, no. 1, pp. 1–14, 2011.

[2] D. R. Obando, "From deterioration modeling to remaining useful life control: a comprehensive framework for post-prognosis decision-making applied to friction drive systems," Ph.D. dissertation, Université Grenoble Alpes, 2018.

[3] 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*, vol. 42, no. 1-2, pp. 314–334, 2014.

[4] E. M. Brakni, "Réseaux de neurones artificiels appliqués à la méthode électromagnétique transitoire infinitem," Ph.D. dissertation, Université du Québec en Abitibi-Témiscamingue, 2011.

[5] Wikistat, "neural network and introduction to deep learning— wikistat," 2016, available Janury 21st 2016, 17pages, pdf format. [Online]. Available: https://www.math.univ-toulouse.fr/besse/Wikistat/pdf/st-m-hdstat-rnn-deeplearning.pdf

[6] F. Chollet, *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. MITP-Verlags GmbH & Co. KG, 2018.

[7] G. S. Babu, P. Zhao, and X.-L. Li, "Deep convolutional neural network based regression approach for estimation of remaining useful life," in *International conference on database systems for advanced applications*. Springer, 2016, pp. 214–228. [Online]. Available: https://doi.org/10.1007/978-3-319-32025-0_14.

[8] J.-R. Jiang and C.-K. Kuo, "Enhancing convolutional neural network deep learning for remaining useful life estimation in smart factory applications," in *2017 International Conference on Information, Communication and Engineering (ICICE)*. IEEE, 2017, pp. 120–123. [Online]. Available: https://doi.org/10.1109/ICICE.2017.8478928.

[9] J. Zhu, N. Chen, and W. Peng, "Estimation of bearing remaining useful life based on multiscale convolutional neural network," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 4, pp. 3208–3216, 2018. [Online]. Available: https://doi.org/10.1109/TIE.2018.2844856.

[10] 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. [Online]. Available: https://doi.org/10.1016/j.ress.2017.11.021

[11] L. Jayasinghe, T. Samarasinghe, C. Yuenv, J. C. N. Low, and S. S. Ge, "Temporal convolutional memory networks for remaining useful life estimation of industrial machinery," in *2019 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2019, pp. 915–920, online available: http://arxiv.org/abs/1810.05644.

[12] A. Al-Dulaimi, S. Zabihi, A. Asif, and A. Mohammadi, "A multimodal and hybrid deep neural network model for remaining useful life estimation," *Computers in Industry*, vol. 108, pp. 186–196, 2019.

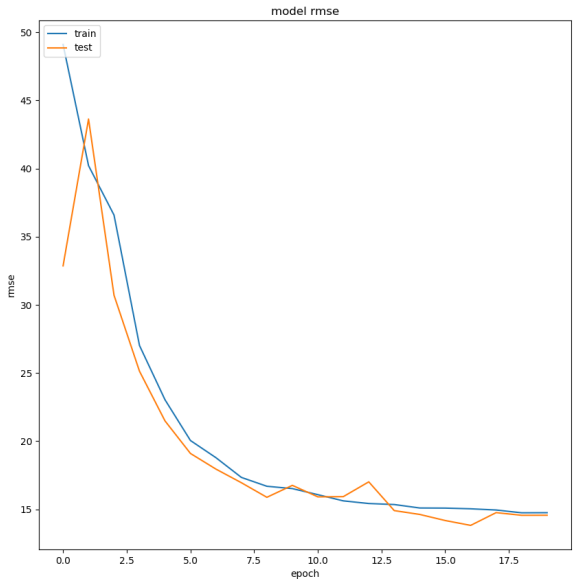

Fig. 3. RMSE values: learning curve of FD001 in the training and test processes.

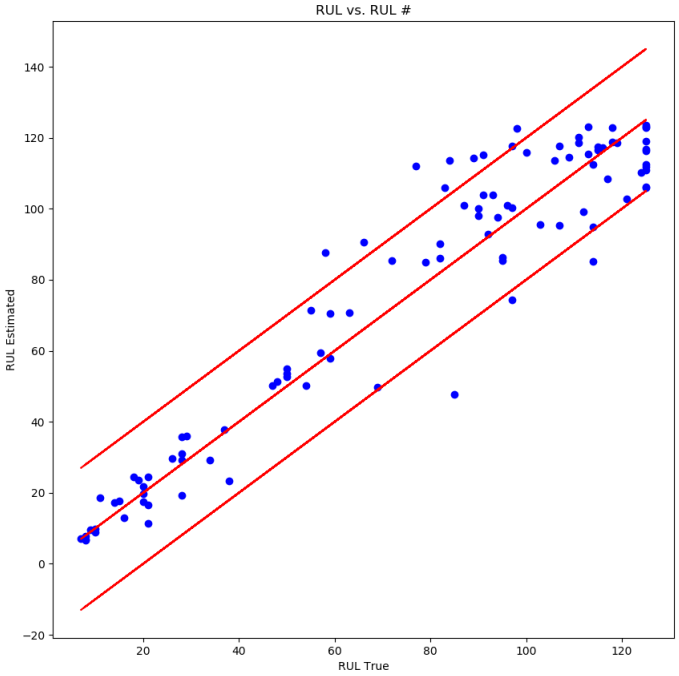

Fig. 4. Comparison between predicted and actual RUL values for testing engines.

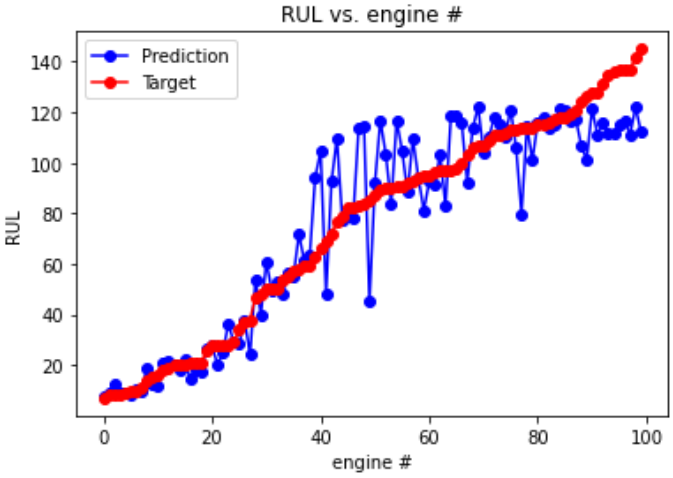

Fig. 5. RUL predictions for the testing engins in FD001

[13] A. Al-Dulaimi, A. Asif, and A. Mohammadi, "Noisy parallel hybrid model of nbgru and ncnn architectures for remaining useful life estimation," *Quality Engineering*, vol. 32, no. 3, pp. 371–387, 2020.

[14] R. Zhao, R. Yan, J. Wang, and K. Mao, "Learning to monitor machine health with convolutional bi-directional lstm networks," *Sensors*, vol. 17, no. 2, p. 273, 2017.

[15] P. Malhotra, V. TV, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Multi-sensor prognostics using an unsupervised health index based on lstm encoder-decoder," *arXiv preprint arXiv:1608.06154*, 2016.

[16] R. Zemouri, Z. Al Masry, I. Remadna, S. L. Terrissa, and N. Zerhouni, "Hybrid architecture of deep convolutional variational auto-encoder for remaining useful life prediction," in *Proceedings of the 30th European Safety and Reliability Conference and the 15th Probabilistic Safety Assessment and Management Conference*, F. D. M. Piero Baraldi and E. Zio, Eds., Copyright c ESREL2020-PSAM15 Organizers. Venice, Italy: Research Publishing Services, Singapore, 2017, pp. 3591–3598. [Online]. Available: https://www.rpsonline.com.sg/proceedings/esrel2020/html/4876.xml

[17] H. Liu, Z. Liu, W. Jia, and X. Lin, "A novel deep learning-based encoder-decoder model for remaining useful life prediction," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8. [Online]. Available: https://doi.org/10.1109/IJCNN.2019.8852129

[18] H. Wang, M.-j. Peng, Z. Miao, Y.-k. Liu, A. Ayodeji, and C. Hao, "Remaining useful life prediction techniques for electric valves based on convolution auto encoder and long short term memory," *ISA transactions*, vol. 108, pp. 333–342, 2021. [Online]. Available: https://doi.org/10.1016/j.isatra.2020.08.031.

[19] A. Al-Dulaimi, S. Zabihi, A. Asif, and A. Mohammadi, "Hybrid deep neural network model for remaining useful life estimation," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3872–3876.

[20] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[21] W. Yu, I. Y. Kim, and C. Mechefske, "Remaining useful life estimation using a bidirectional recurrent neural network based autoencoder scheme," *Mechanical Systems and Signal Processing*, vol. 129, pp. 764–780, 2019.

[22] C. Yin, S. Zhang, J. Wang, and N. N. Xiong, "Anomaly detection based on convolutional recurrent autoencoder for iot time series," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[23] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International conference on artificial neural networks*. Springer, 2011, pp. 52–59.

[24] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring," *Mechanical Systems and Signal Processing*, vol. 115, pp. 213–237, 2019.

[25] Y. Liu and N. G. R. Center, *User's guide for the commercial modular aero-propulsion system simulation (C-MAPSS)*, ser. NASA technical memorandum. National Aeronautics and Space Administration, Glenn Research Center, 2012. [Online]. Available: https://books.google.dz/books?id=MhGXnQAACAAJ

[26] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *2008 interna-*

tional conference on prognostics and health management. IEEE, 2008, pp. 1–9.

[27] D. Mishra. Regression: An explanation of regression metrics and what can go wrong. [Online]. Available: https://towardsdatascience.com/regression-an-explanation-of-regression-metrics-and-what-can-go-wrong-a39a9793d914

[28] C. Staudemeyer, Ralf Omlin, "Evaluating performance of long short term memory recurrent neural networks on intrusion detection data," in *CM International Conference Proceeding*, ser. 218, 2013.

[29] R. C. Staudemeyer and C. W. Omlin, "Evaluating performance of long short-term memory recurrent neural networks on intrusion detection data," in *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, 2013, pp. 218–224.

[30] T. Xia, Y. Song, Y. Zheng, E. Pan, and L. Xi, "An ensemble framework based on convolutional bi-directional lstm with multiple time windows for remaining useful life estimation," *Computers in Industry*, vol. 115, p. 103182, 2020.

[31] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[32] I. Remadna, S. L. Terrissa, R. Zemouri, S. Ayad, and N. Zerhouni, "Leveraging the power of the combination of cnn and bi-directional lstm networks for aircraft engine rul estimation," in *2020 Prognostics and Health Management Conference (PHM-Besançon)*. IEEE, 2020, pp. 116–121.

[33] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, "Long short-term memory network for remaining useful life estimation," in *2017 IEEE international conference on prognostics and health management (ICPHM)*. IEEE, 2017, pp. 88–95.

[34] Y. Zhang, R. Xiong, H. He, and M. G. Pecht, "Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 7, pp. 5695–5705, 2018.

[35] H. Yang, F. Zhao, G. Jiang, Z. Sun, and X. Mei, "A novel deep learning approach for machinery prognostics based on time windows," *Applied Sciences*, vol. 9, no. 22, p. 4813, 2019.

[36] B. Wang, Y. Lei, N. Li, and T. Yan, "Deep separable convolutional network for remaining useful life prediction of machinery," *Mechanical Systems and Signal Processing*, vol. 134, p. 106330, 2019.

[37] C. Zhao, X. Huang, Y. Li, and M. Yousaf Iqbal, "A double-channel hybrid deep neural network based on cnn and bilstm for remaining useful life prediction," *Sensors*, vol. 20, no. 24, p. 7109, 2020.

[38] J. Narwariya, P. Malhotra, V. TV, L. Vig, and G. Shroff, "Graph neural networks for leveraging industrial equipment structure: An application to remaining useful life estimation," *arXiv preprint arXiv:2006.16556*, 2020.

[39] Y. Jiang, Y. Lyu, Y. Wang, and P. Wan, "Fusion network combined with bidirectional lstm network and multiscale cnn for remaining useful life estimation," in *2020 12th International Conference on Advanced Computational Intelligence (ICACI)*. IEEE, 2020, pp. 620–627.

[40] I. Remadna, L. S. Terrissa, S. Ayad, and N. Zerhouni, "Rul estimation enhancement using hybrid deep learning methods," *International Journal of Prognostics and Health Management*, vol. 12, no. 1, 2021.