RUL Estimation Enhancement using Hybrid Deep Learning Methods

Ikram Remadna [1], Labib Sadek Terrissa [1], Soheyb Ayad [1], and Noureddine Zerhouni [2]

[1] *LINFI Laboratory University of Biskra, Biskra,07000 Algeria*
ikram.remadna@univ-biskra.dz, ikram.remadna@outlook.com
terrissa@univ-biskra.dz, terrissalabib@gmail.com
s.ayad@univ-biskra.dz, soheyb.ayad@gmail.com

[2] *Femto-ST Institute AS2M Departement Besançon, Besançon, 25000 ,France*
zerhouni@ens2m.fr

ABSTRACT

The turbofan engine is one of the most critical aircraft components. Its failure may introduce unwanted downtime, expensive repair, and affect safety performance. Therefore, It is essential to accurately detect upcoming failures by predicting the future behaviour health state of turbofan engines as well as its Remaining Useful Life (RUL). The use of Deep Learning (DL) techniques to estimate RUL has seen a growing interest over the last decade. However, hybrid DL methods have not been sufficiently explored yet by researchers.
In this paper, two-hybrid methods proposed to enhance the RUL estimation by combining Convolutional Auto-encoder (CAE), Bi-directional Gated Recurrent Unit (BDGRU), Bi-directional Long-Short Term Memory (BDLSTM), and Convolutional Neural Network (CNN). The results indicate that the proposed hybrid methods significantly outperform the robust predictions in the literature.

1. INTRODUCTION

As an essential part of the aircraft, the turbofan engine is a complex and sophisticated system; its safety and reliability are indispensable. Any unexpected breakdown in the engine before its overhaul will lead to a severe accident that may cost millions in lost human lives, pollution, costly repair, etc (Saxena, Goebel, Simon, & Eklund, 2008). Therefore, maintaining aircraft engine reliability presents a challenge to reduce engine downtime and maintenance costs without jeopardizing safety and ensuring engine availability. According to statistics, aircraft engine maintenance costs are approximately 70% of their whole life cycle costs (Guo, 2015). Consequently, it is essential to integrate an optimal maintenance strategy that detects upcoming degradation by predicting the engines future health state, preventing unplanned downtime, and reducing maintenance costs.

In turbofan engine maintenance, several intelligent maintenance strategies are evolved from traditional ones (Gouriveau, Medjaher, Ramasso, & Zerhouni, 2013). Traditional maintenance is either reactive way (fixation or replacement of engine component after the detection of its breakdown) or proactive way (controlling the scheduling maintenance tasks based on the assumption of a certain level of performance degradation whether maintenance is essential or not). Currently, both ways are inefficient and unable to eradicate faults or to conduct them (Ding & Kamaruddin, 2015). An intelligent maintenance strategy, referred to as predictive maintenance, coordinates the scheduling maintenance tasks based on the fault diagnosis, fault prognosis, and RUL estimation.

Prognostics and Health Management (PHM) has appeared as a predictive maintenance process, which offers several advantages. Its main functions are fault detection, fault isolation, also failure prognostics that allows predicting RUL, finally making appropriate maintenance decision (Atamuradov, Medjaher, Dersin, Lamoureux, & Zerhouni, 2017). Predicting the RUL with high accuracy plays a critical role in the PHM process since its inaccurate estimation may cause unexpected catastrophic failures. Recently, various RUL prediction methods have been proposed, which can be categorized into three main approaches (Gouriveau, Medjaher, & Zerhouni, 2017): (1) physics model-based approach, (2) data-driven approach, and (3) hybrid approach. The physics model-based approach uses the mathematical model that can be a set of differential or algebraic equations, which are very useful to predict RUL in cases where the failure data available are insufficient. This approach requires extensive physical background and knowledge. However, the data-driven approach is used to model

https://doi.org/10.36001/IJPHM.2021.v12i1.2378

the degradation and estimate the RUL for the machine with enough failure data. The ease of collecting the monitoring data of many industrial systems has motivated many researchers to use data-driven models in estimating the RUL. Moreover, the hybrid approach integrates physics model-based and data-driven approaches to estimate its RUL.

Extracting performance degradation features from multi sensor data is a critical technical problem for complex systems as data grows high dimensional, which significantly impacts the prediction performance. Unfortunately, the popular way of designing features manually for complicated domains requires much human labour, and most information can be lost, and performance can not be guaranteed (Mierswa & Morik, 2005). Additionally, the traditional popular feature extraction methods as Principal Component Analysis (PCA) (Demšar, Harris, Brunsdon, Fotheringham, & McLoone, 2013) and Linear Discriminant Analysis (LDA) (Sharma & Paliwal, 2015) are unsupervised methods based on a projection of the original high-dimensional space to a low-dimensional space by maximizing the correlation between the data. However, they suffer from being based on linear projection. Therefore, the non-linear features extraction method has been exploited to learn the useful and the low-dimensional features, such as ISOMAP (Tenenbaum, De Silva, & Langford, 2000), and Locally Linear Embedding (LLE) (Roweis & Saul, 2000). However, the major problem with these techniques is that they have a predetermined way of extracting local data relationships among data samples, which may be inaccurate in a low-dimensional space.

Since 2006, a new branch of machine learning methods called Deep Learning was introduced to treat highly non-linear and varying data in their raw form without any human labour (LeCun, Bengio, & Hinton, 2015). DL is characterized by a deep hierarchical structure where several processing layers are stacked to learn automatically high-level representations from large-scale data that are ultimately useful for improving the prediction or classification. Various deep learning algorithms, including Long-Short Term Memory (LSTM), Deep Neural Network (DNN), Auto-Encoders (AE), Deep Belief Networks (DBN), and Convolutional Neural Network, have been proposed and outperformed conventional machine learning algorithms. Today, various researchers have shown the success of these DL architectures in many fields, including computer vision (Voulodimos, Doulamis, Doulamis, & Protopapadakis, 2018), natural language processing (Collobert & Weston, 2008), the application of machine health monitoring (Zhao et al., 2019), diagnostics in healthcare (Belaala, Bourezane, Terrissa, Al Masry, & Zerhouni, 2020).

The most current RUL estimation architectures incorporate only a single method such as CNN, DNN, or LSTM. Accordingly, the idea of the hybrid method and the application of a parallel multi-model emerged to leverage the power of different models that capture various information at different time intervals and ultimately achieving more accurate predictions, which was not previously addressed. Recently, CNN has achieved promising results in RUL prediction, which was exploited to capture spatial features without considering the time-series correlation to data. Conversely, BDLSTM or BDGRU is capable of capturing bi-directional temporal dependencies features from sensors data. This paper proposed two-hybrid methods based on these promising architectures. The first proposed hybrid method uses CAE as a feature extractor combined with two temporal modelling tools simultaneously in a parallel way (referred to as BDGRU-BDLSTM). The second promising hybrid architecture blends the CNN and BDGRU simultaneously to capture local and temporal features directly from raw sensory data instead of just using CNN for feature extraction (referred to as CNN-BDGRU). The public NASA's C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) dataset is applied to verify the superiority and effectiveness of the proposed methods. The principal contributions of this research are summed up as follows:

- An end-to-end deep network structures for RUL predictions of a turbofan engine are proposed and discussed in this paper to provide a comprehensive comparison of different types of deep learning-driven approaches.
- Multi-model helps to leverage the power of different models in which two DL models are integrated in a parallel manner that capture various information at different time intervals.
- Extracting succinct and useful degradation features using the unsupervised Convolutional Auto-Encoders method from multi-sensor data with complex correlations.
- A detailed case study demonstrating the superiority of proposed hybrid models is presented.

The paper's remainder is structured as follows: Section 2 reviews recent applications of deep learning models on the C-MAPSS dataset. Section 3 describes the proposed hybrid methods for RUL estimation of a turbofan engine. The experimental results and discussions that demonstrate the effectiveness and superiority of the proposed hybrid models are considered in Section 4. Finally, we close the study with conclusions and future work in section 5.

2. LITERATURE REVIEW

The C-MAPSS dataset has been presented and applied to evaluate various effective DL methods for aircraft engine RUL estimation in recent years. In this section, we survey works that applied to the C-MAPSS dataset and have leveraged deep learning methods to tackle the task of RUL estimation. The selected works are presented in the following excerpts, which either applied the CNN, LSTM, DNN using auto-encoders.

2.1. CNN

Within the deep learning architecture, the first implementation of CNN for RUL estimation of aircraft engines proposed in (Babu, Zhao, & Li, 2016), where the input data is segmented into sliding windows and afterwards normalized. The CNN structure's ability to learn a higher-level abstract representation along with the multi-channel time series through its convolutional and average-pooling layers is shown. A linear regression layer is attached to the top layer to perform RUL predictions. The results showed the superiority and effectiveness of the CNN model over other machine learning models such as the Multilayer Perceptron (MLP), the Support Vector Machine (SVM), and the Relevance Vector Machine (RVM). In a similar study, Li et al. (Li, Ding, & Sun, 2018) proposed a novel deep CNN-based approach for RUL forecasts of aircraft turbofan engines. The authors employ a time window strategy for data processing for improving the feature extraction via deep CNN. The normalized sensors data are directly utilized as the model inputs. Besides, they use the dropout technique to prevent overfitting. This model achieves the most accurate estimation of RUL and the lowest Root Mean Square Error (RMSE) than 13 other data-driven methods. The authors also highlighted that optimum performance is achieved through 5 convolution layers and a time window length of 30.

2.2. RNN and its variants

The Recurrent Neural Network (RNN) retains internal memory to process sequential data. However, RNNs had the vanishing gradient problem arising in long sequence input, which cannot keep the previous information, except the latest one. To handle this issue, Zheng et al. (Zheng, Ristovski, Farahat, & Gupta, 2017) suggested an engine RUL prediction method based on deep LSTM, capable of capturing long-range dependencies of different time scales. The model consists of two LSTM layers combined with two Feed-forward Neural Network (FNN) layers and the output layer. Results reveal that deep LSTM outperforms CNN presented by other researchers (Babu et al., 2016) based on RMSE. Similarly, Hsu et al. (Hsu & Jiang, 2018) proposed an LSTM to address the RUL prediction problem for turbine engines, which is able effectively to extract temporal dependencies from the historical data. Liao et al. (Liao, Zhang, & Liu, 2018) have used LSTM relying on the bootstrap procedure for uncertainty estimation of RUL. The bootstrap method is a good solution to obtain uncertainty prediction without any sensor data distribution. The proposed approach achieved higher accuracy compared to CNN and LSTM discussed in (Babu et al., 2016) and (Zheng et al., 2017), respectively. Additionally, Yuan et al. (Yuan, Wu, & Lin, 2016) proposed an LSTM to determine the fault location and estimate the RUL of the aero-engine in the cases of complicated operations, hybrid failures, and strong noises. More recently, Wu et al. (Wu et al., 2020) proposed a new deep LSTM approach for discovering the hidden long-term dependencies among sensor time-series signals to predict RUL. The grid search was also applied to tune the hyperparameters, thereby obtaining the best network structure. This method showed enhanced performance compared to other methods in the literature.

Another variant of LSTM was used in (Wang, Wen, Yang, & Liu, 2018) is Bi-directional LSTMs that can learn the bi-directional temporal dependencies from sensor data for Aircraft Engine RUL estimation. It can capture long-range information in both future (forward) and past (backward) contexts of the input sequence simultaneously. In another study, a new bi-directional LSTM model was presented in (Zhang, Wang, Yan, & Gao, 2018) for identifying the system degradation performance and subsequently predicting RUL. The proposed model consists of two BDLSTM layers and achieved promising results compared to LSTM, bi-directional RNN (BDRNN), MLP, CNN reported by (Babu et al., 2016). Another main variant of RNN that is recently utilized for RUL estimation and enhanced LSTM-model with few parameters, Gated Recurrent Unit (GRU) is presented by (Chen, Jing, Chang, & Liu, 2019). The authors proposed a new approach for RUL estimation of a nonlinear degradation process, using Kernel PCA (KPCA) as the first phase for dimensionality reduction and nonlinear feature extraction. The second phase uses GRU to prevent the problem of long-term dependency and allows each recurrent unit to adaptively extract dependencies of different time scales.

2.3. DNN using auto-encoders

In addition to the CNN and RNN architectures, AE is another main structure that is essentially a feature extractor for reducing data monitoring condition performed in an unsupervised manner. Many studies have shown the leverage of using AE alongside another machine learning method for estimating the RUL of a turbofan engine (Song, Shi, Chen, Huang, & Xia, 2018)(Ma, Su, Zhao, & Liu, 2018). Song et al. (Song et al., 2018) proposed a new hybrid model integrating the advantages of AE and bidirectional LSTM to enhance the RUL's prediction accuracy. The main idea is that the encoding part of AE (bottleneck) acts as input for the BDLSTM to produce the expected output. The results demonstrate that the combination of AE and BDLSTM outperformed the other methods, such as MLP, CNN, LSTM, BDLSTM and Autoencoder-LSTM. In this work, Ma et al. (Ma et al., 2018) also proposed a novel end-to-end deep architecture based on a stacked sparse Autoencoder (SAE) and logistic regression. This study utilized the grid search procedure to optimize the hyper-parameters of the SAE model.

Inspired by these previous studies, the idea of the hybrid approach and applying a parallel multi-model emerged to leverage the power of different methods, which have high potentials to boost prognostic accuracy instead of incorporating

only on a single model such as CNN, DNN, or LSTM. Accordingly, capitalizing on the recent success of DL, this paper presents a framework driven by an end-to-end ML system that introduces two new hybrid RUL prediction approaches to capture various information through different time intervals. The previous studies have shown the advantage of using AE alongside another machine learning method for estimating the RUL of the turbofan engine. Traditional AE used a fully connected layer reported in the literature, whereas the CAE model has a promising ability for feature extraction and dimensionality reduction through convolutional layers. Aligning with Convolutional Auto-Encoders's power, the first promising proposed hybrid model adopts CAE in aero-engine prognostic problem to extract automatically useful features with a high-level of abstractions. These CAE features serve as inputs to train the two temporal modeling tools simultaneously in a parallel manner (referred to as BDLSTM path and BDGRU path) that can capture more robust features and eventually predict the RUL. Although CAE has been applied on different tasks, this is the first use of CAE in RUL's estimation problem for engine turbofan.

For a comprehensive comparison, the second hybrid architecture proposed differently from what has been reported in the literature, which consists of CNN and BDGRU models simultaneously in parallel paths to capture local and temporal features directly from raw sensory data, instead of just using CNN. The outputs from both paths (CNN and BDGRU) are concatenated to obtain the target RUL. The GRU has appeared as an enhanced LSTM model with few parameters for improving the training phase's speed and model performance. Besides, we used a BDGRU for the bi-directional temporal feature extraction and preventing the long-term dependency problem. The superiority of the two proposed hybrid models is demonstrated using the public NASA's C-MAPSS dataset and by comparison with all its counterparts and the most robust results in the literature.

3. PROPOSED HYBRID DEEP LEARNING MODELS

This section introduces the relevant hybrid deep learning approaches proposed in this research for the RUL prediction of an aircraft engine. Figure 1 describes the proposed framework for RUL estimation that comprises two main stages. In the training stage, which is completely offline, the obtained historical data from sensors flow through the components of the training stage, ultimately the degradation model for the RUL estimation is constructed based on deep learning methods. In the prediction stage, conducted online, the obtained current data is stored in the dataset and processed to obtain normalized sequence data, where the trained model is applied to predict the RUL. Based on the RUL values, a maintenance action will be applied to the system at the exact scheduled moment.

3.1. Problem Formulation

Considering that there are N machines of the same type, such as the turbofan engine, each engine contains T_i run to machine end of life (cycles) collected by multiple sensors. Mathematically, the whole data can be defined as :

$$Dataset = \{(X^i, Y^i)\},\ i = 1, 2, 3, ..., N \tag{1}$$

Thus, X^i denotes the gathered sensor measurements matrix of an engine in which Y^i corresponds to the equipment operation cycles, as shown in Eq.2 and Eq.3, respectively.

$$X^i = [x_1, x_2, x_t, ..., x_{T_i}] \in R^{m \times T_i} \tag{2}$$

$$Y^i = [y_1, y_2, ..., y_{T_i}] \in R^{1 \times T_i} \tag{3}$$

Where T_i is the total operation cycles of the i-th engine and $x_t = [x_t^1, x_t^2, ..., x_t^m] \in R^{m \times 1}$ is an m-dimensional vector of sensor measurements at time t.

RUL can be calculated between the current time (y_t) after degradation (td) has been detected and the failure time (T). The RUL can be described as follows (Javed, Gouriveau, & Zerhouni, 2017).

$$RUL_t^i = \{T_i - y_t^i\},\ t = 1, 2, ..., T_i \tag{4}$$

To address the non-linearity function, deep learning methods are proposed (φ) in this paper. Let X^i denote its input, and the observed RUL is as its output.

$$\check{RUL}^i = \varphi(x^i, RUL^i) \tag{5}$$

To minimize the error between the predicted RUL value and the observed target RUL at time t.

$$Minimize : \{\check{RUL}_t^i, RUL_t^i\} \tag{6}$$

3.2. Convolutional Auto-encoder with BDGRU-BDLSTM Hybrid Model

As illustrated in Figure 2, our proposed hybrid model comprises of two main stages: The first one is CAE, which is used to automatically extract performance degradation features while lowering the dimension of multiple sensors in an unsupervised manner. The second stage is the temporal modeling tool, which combines BDGRU and BDLSTM models simultaneously and in a parallel way to provide the RUL's estimation. The full details of the two main stages are described as follows.

A. **Stage 1: CAE module**

The CAE architecture consists of two parts, an encoder and a decoder, two symmetrical and reversed structures. The encoder network part comprises six convolution layers with the same filter size (10×1) and one max-pooling layer. Precisely in this work, two-pairs of convolution

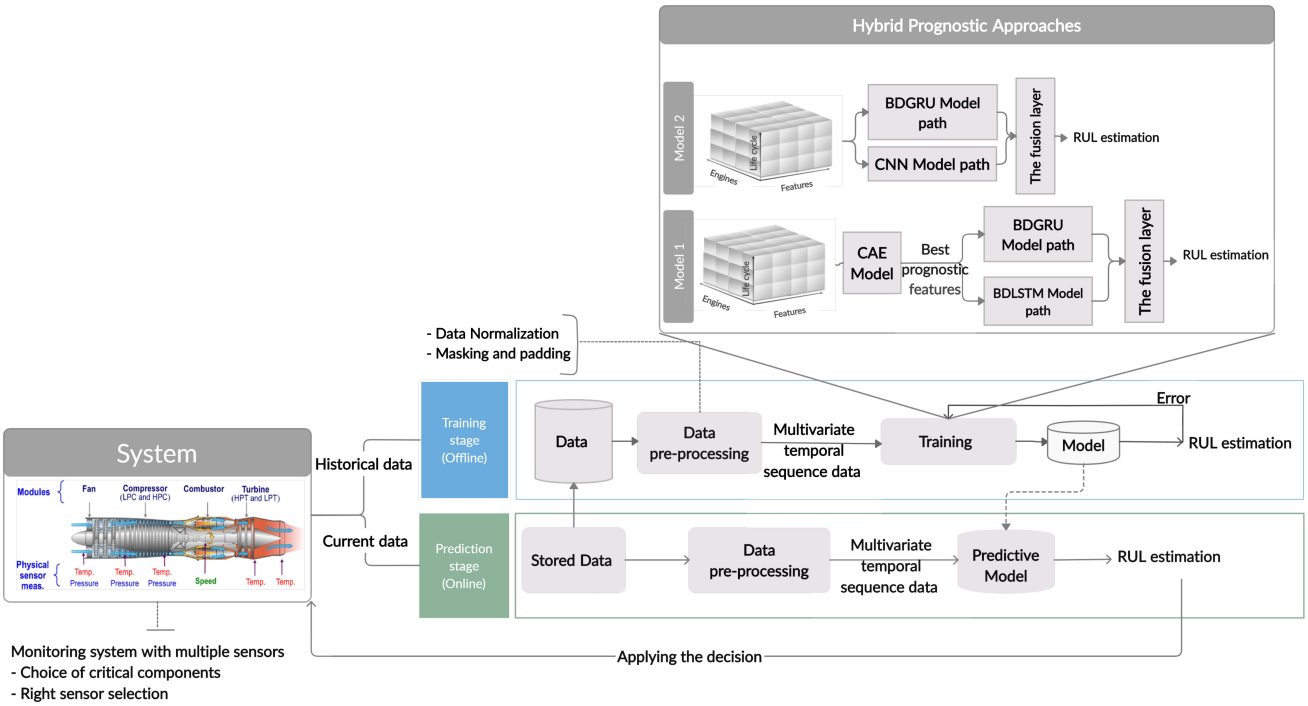

Figure 1. The proposed framework for RUL estimation.

layers are stacked, where the number of filters is set to 8 and 18, followed by one Max-pooling layer with filter size (1×2). The third and fourth CNN layers consist of 32 and 64 filters, respectively. After every two-pairs of convolution layers, a dropout layer is added to reduce overfitting and avoid repeatedly extracting the identical features. To obtain a unique feature map, the final convolution layer's filter defined by one. All convolution layers used Rectified Linear Unit (ReLU) as the activation function. Furthermore, the zero-padding operation is used to maintain the feature map unaltered.
The operations of un-pooling and de-convolution are used in the decoder part of CAE to reconstruct the input instead of convolution and max-pooling operations used in the encoder part.

B. **Stage 2: BDGRU-BDLSTM hybrid module**

The useful features learned from CAE are used as input to the multi-model structure that can maintain good generalization performance. The proposed multi-model is a temporal modeling tool, which is based on the combination of BDGRU and BDLSTM models simultaneously. This combination aims to obtain more robust features and eventually predict the RUL. Both paths (BDGRU and BDLSTM) share the same configuration where two layers are stacked with 50 nodes. The layers use the hyperbolic tangent (tanh) as an activation function. A dropout technique is applied with a rate of 0.25 per layer.

At the end to estimate the RUL, the outputs from both paths are concatenated and will be inserted into a fully connected layer; this layer has one neuron and uses the exponential activation function.

3.3. CNN-BDGRU Hybrid Model

As shown in Figure 3, the proposed hybrid model is based on the combination of CNN and BDGRU in a parallel manner for regression. The CNN path acts as a spatial feature extractor, while simultaneously, the BDGRU path is utilized for the bidirectional temporal features extraction. Although there is no correlation between the two pathways, their outputs are concatenated to obtain the RUL's overall prediction.
Specifically, the BDGRU structure is designed to handle each sequence data in two directions, through the GRU cells forward for prediction and backward direction for smoothing the prediction and relieving the noise impact. Below, we detail the structure of three major components of our hybrid model:

A. **The CNN path**

In our proposed model, CNN is exploited to capture spatial features by stacking the convolutional kernels. CNN is composed of five convolution layers, which have the same filter size (10×1). The first and second CNN layers consist of 18 filters, while the third and fourth layers contain the same number of 32 filters. To obtain a unique feature map, the final convolution layer is used with a single filter to fuse all the previous feature maps. The

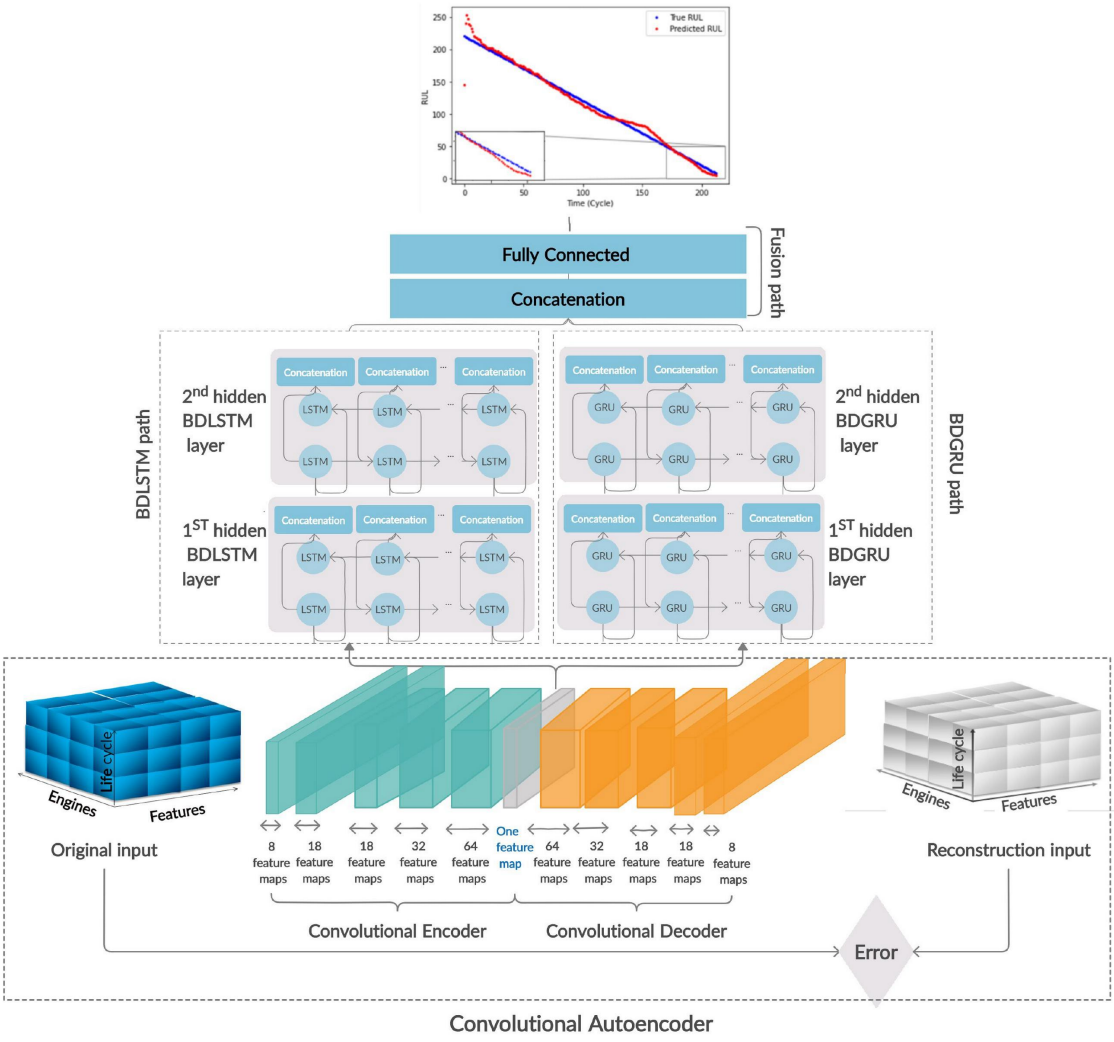

Figure 2. The first proposed hybrid model based on CAE with BDGRU-BDLSTM.

ReLU is applied along with zero-padding for all convolution layers. In this way, a high-level representation is obtained for each raw collected features.

B. The BDGRU path

The BDGRU is selected to learn the long-rang dependencies of features. Through this path, both forward direction and backward direction are computed in two separate GRUs independently. Their outcomes are fused and distributed to the next layer. Two BDGRU layers are stacked within the same configuration as the first proposed method is used.

Besides, the BDGRU and GRU share the same cell architecture that allows addressing the vanishing gradient problem. Furthermore, the hidden state of the BDGRU cell is expressed as follows:

$$h_T = (\overrightarrow{h_T} \bigotimes \overleftarrow{h_T}) \tag{7}$$

Where $\rightarrow$ and $\leftarrow$ symbolize forward and backward process, respectively.

C. The fusion path

The final prediction at each time step is achieved by concatenating the outputs from both paths (CNN path and BDGRU path). This fusion layer has one neuron and uses the exponential activation function.

4. EXPERIMENT STUDY

In this section, the performance of the deep learning-driven prognosis approaches is evaluated on a prognostic benchmarking problem. First, the C-MAPSS simulated turbofan engine dataset descriptions are presented in Section 4.1, followed by these main sections including data pre-processing, evaluation metric, prediction procedure. Finally, the details of the results and comparisons with several architectures to provide a comprehensive examination of the proposed models are discussed

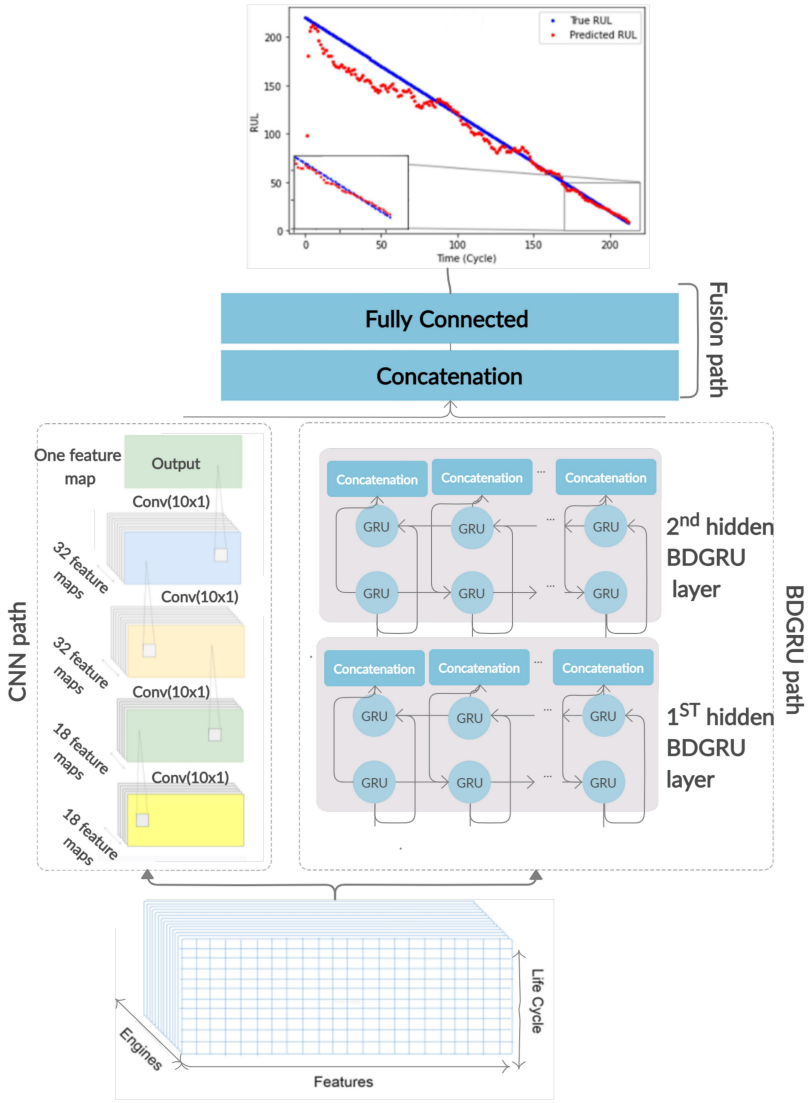

Figure 3. The second proposed hybrid model based on CNN and BDGRU.

in Section 4.5.

Besides, all experiments are carried out on a personal computer with Intel Core i5- 5200U (2.20 GHz) Central Processing Unit (CPU), 06 Go Random Access Memory (RAM), the Microsoft Windows operation system 64 bits. The programming language is python 3.5 with deep learning library Keras.

4.1. NASA C-MAPSS dataset description

NASA's C-MAPSS datasets are immensely used by scientists in the field of RUL prediction (Saxena & Goebel, 2008). The main gas turbine engine modules include the fan, low-pressure compressor (LPC), high-pressure compressor (HPC), high-pressure turbine (HPT), low-pressure turbine (LPT), and nozzle, as shown in Figure 1. The C-MAPSS datasets represent the gas turbine engines deterioration. The four fleets are into four sub-datasets in C-MAPSS with varying number of operating and fault conditions. Each sub-dataset is separated into training and testing sets, as seen in Table 1.

4.2. Data pre-processing

The subsets FD001 and FD003 exhibit constant sensor measurements and three operational settings throughout the lifetime of the engine; which could not be useful for RUL estimation. However, all three operational sensor settings and all sensor measurements can provide useful information about the deterioration of a turbofan engine in FD002 and FD004. Consequently, unlike works (Li et al., 2018), (Ma et al., 2018) (Wang et al., 2018), which excluded the three operational settings and selected 14 sensors out of the 21 sensors. The pro-

posed methods, all three sensor settings and all sensor measurements are picked as input features for all sub-datasets. The goal is to avoid designing features manually by proposing flexible models of an End-to-End ML system using Deep Learning.

It is essential to prepare the data before training the models. Therefore, the data normalization, the masking, and the padding phase are used in this study.

Table 1. The C-MAPSS dataset.

Data set	FD001	FD002	FD003	FD004
Train trajectories	100	260	100	248
Test trajectories	100	259	100	249
Operating conditions	1	6	1	6
Fault conditions	1(HPC degradation)	1(HPC degradation)	2(HPC degradation, fan degradation)	2(HPC degradation, fan degradation)
Maximum life span	362	378	525	543

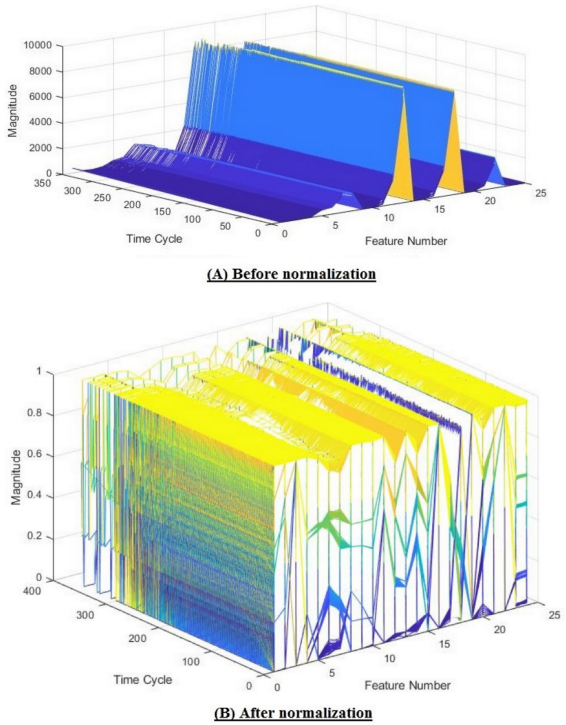

Figure 4. An illustration of FD002 testing data before and after normalization.

4.2.1. Data Normalization

According to the differentiation issue of features range scales, several normalization methods have been proposed to ensure the same range scale of all features (Patro & Sahu, 2015). The Min-Max normalization given in Eq.8, is used to map the raw features within the range of [0,1].

$$x_i' = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \tag{8}$$

Where x_i is the time sequence of the i^{th} sensor measurements, Min and Max are the minimum and maximum values in x_i given its range, and the x_i' is the normalization input data. Figure 4 shows an illustration of FD002 testing data before and after normalization.

4.2.2. Masking and padding

The engines have varying length cycles, and hence, the shorter sequences than the maximum length of cycles in the whole dataset are padded with zeros to obtain the same length. Consequently, mask zero is used in the training phase to record if the time step already exists or just padding.

4.3. Evaluation Metric

In this work, the RMSE and scoring function are used for evaluating the model's performance. The formulation of RMSE is defined in Eq.9 to measure the effectiveness of the RUL prediction methods. The RMSE function penalizes both early (underestimate) and late (overestimate) predictions.

$$RMSE = \sqrt{(1/N)\sum_{i=1}^{N}(\Delta_i^2)} \tag{9}$$

where N represents the total number of data samples. $\Delta_i = RUL_i' - RUL_i$, Δ_i is the error between the predicted RUL' value and the true RUL for the i^{th} test samples.
The scoring function adopted by the international conference on PHM data challenge is shown in Eq.10

$$\begin{aligned} S &= \sum_{i=1}^{N} S_i \\ S_i &= \begin{cases} e^{-\Delta_i/13} - 1, & \text{if } \Delta_i<0 \\ e^{\Delta_i/10} - 1, & \text{if } \Delta_i \geq 0 \end{cases} \end{aligned} \tag{10}$$

This scoring function takes into account the impact of the maintenance costs, in which a higher penalty is imposed when the RUL is overestimated. Under this estimation, the maintenance will be scheduled after the appropriate time.

4.4. Prediction procedure

For both proposed methods, first, The C-MAPSS sub-datasets are pre-processed where the data are normalized and padded. Next, the training sub-datasets are split into training and validation sets; 80% of the engines in sub-dataset are randomly

selected for the training, while the remaining 20% are designated as validation set.

4.4.1. CNN-BDGRU Training procedure

The flowchart of the proposed CNN-BDGRU is described in Figure 5. The hybrid model receives as inputs the normalized training set and the RUL values adopted as the target outcomes. The shape of the BDGRU input is a 3D tensor [Min_Batch_size, Time_series_length, Feature_size], where Min_Batch_size is the number of engines in mini-batch size equal to 128. Time_series_length equal to the maximum length of cycles in the whole dataset, and Feature_size equal to 24. On the other side, the shape of the CNN input is a 4D tensor [Min_Batch_size, Time_series_length, Feature_size, 1].

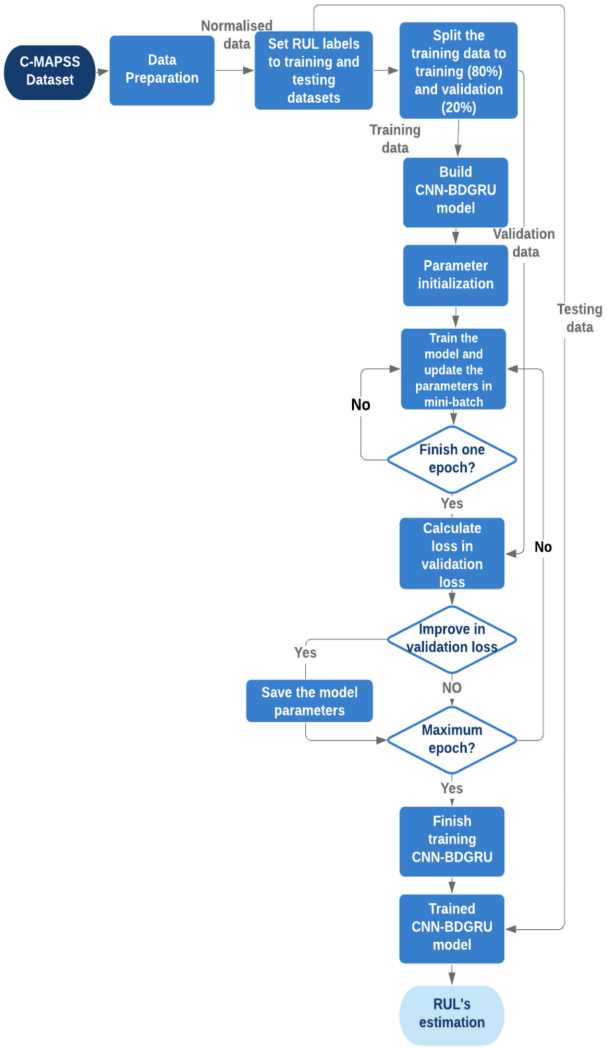

Figure 5. The flowchart of the proposed CNN-BDGRU model.

In the training process, a gradient-based optimization algorithm adjusts the weights in the network based on the minimization of the objective function. Specifically, the Root Mean Square propagation (RMSprop) optimizer method is used to optimize the training model, with the learning rate set at 0.001 to achieve stable convergence (Ruder, 2016). Besides, the Mean Square Error (MSE) serves as the loss function, which is expressed as,

$$MSE = (1/N)\sum_{i=1}^{N}(\Delta_i^2) \tag{11}$$

Where $\Delta_i = RUL_i' - RUL_i$.

The maximum number of training epochs is 3000. For each training epoch, the samples are segmented into mini-batches. To avoid overfitting, the early stopping technique of regularization introduced. Its principal idea is that in the absence of the improvement in performance, the training process is discontinued.

Finally, the testing data samples are fed into the hybrid training model to estimate the RUL and obtaining the RMSE accuracy in the test set.

4.4.2. CAE with BDGRU-BDLSTM Training procedure

The process of training the proposed hybrid CAE with the BDGRU-BDLSTM model consists of two modules: a CAE model as the first phase and the BDGRU-BDLSTM method in the second phase. Both modules used the RMSprop optimizer and the MSE as a cost function. The whole process of training the proposed deep hybrid CAE with the BDGRU-BDLSTM model is summarized in Figure 6.

The whole CAE network is trained in an unsupervised manner that takes the normalized training set as input to reconstruct it; the encoder part represents the more robust deterioration features. The CAE network's weights updated iteratively during training through a gradient-based optimization algorithm based on the minimization of reconstruction errors (MSE), expressed in Eq.11, where $\Delta_i = X_i' - X_i$, Δ_i is the error between the reconstruction X' and the original input X for the i^{th} test samples. Besides, the samples are grouped into mini-batches for each training epoch, with a limit of 1000 training epochs.

After the CAE network's training process, the second step is to train the multi-model (BDGRU-BDLSTM) for the RUL estimation, where the encoder parameters are frozen during the multi-model training step. The extracted CAE features are fed to the multi-model (BDGRU-BDLSTM) as inputs, and the RUL values of the training set are used as the target outputs. Backpropagation through time (BPTT) is the training algorithm applied to update the weights for minimizing the error using RMSprop, with the learning rate set at 0.001. Furthermore, MSE is utilized as the loss function, which is expressed in Eq.11, Where $\Delta_i = RUL_i' - RUL_i$. The total

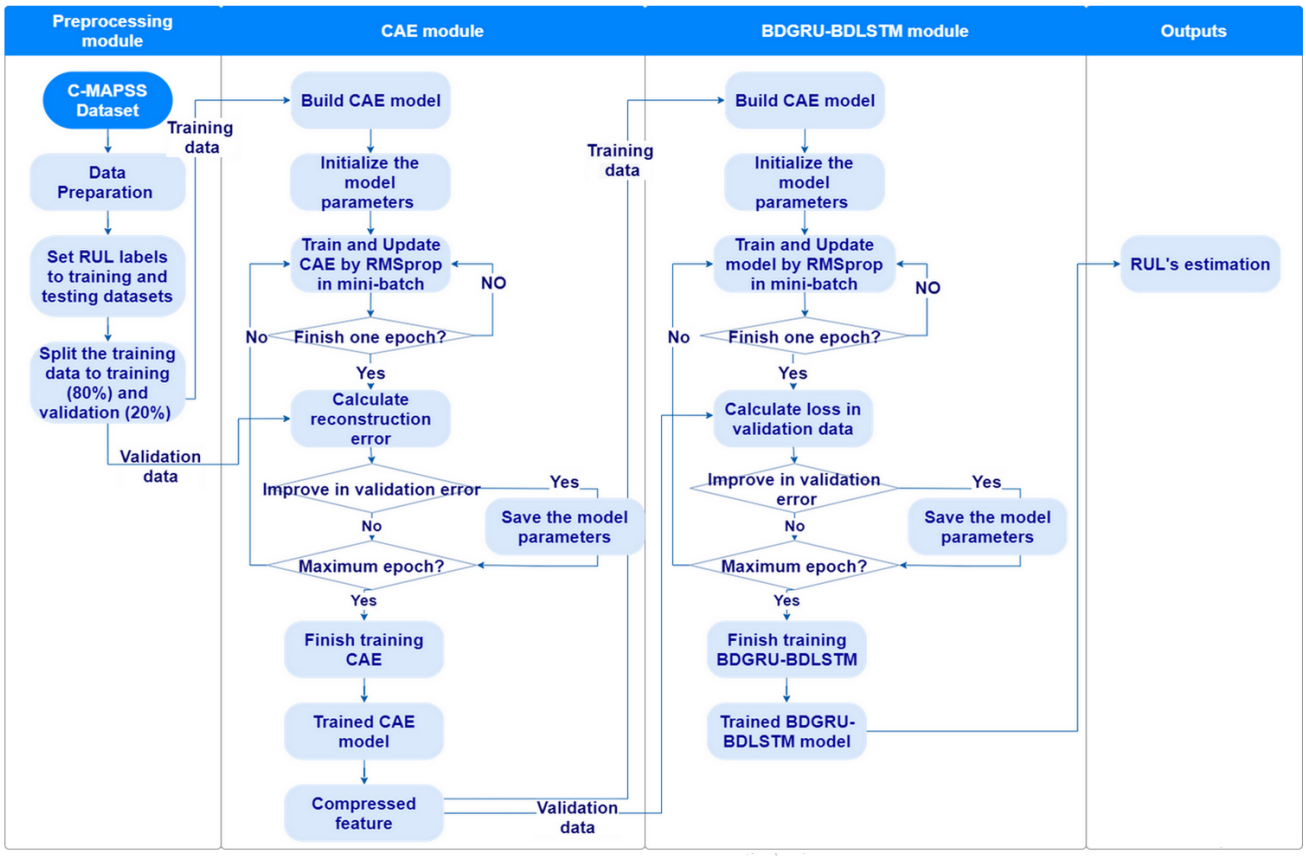

Figure 6. The flowchart of the proposed hybrid model based on CAE and BDGRU-BDLSTM.

number of training epochs is 3000, with the application of the early stopping technique.

Finally, the convolutional encoder is used jointly with the BDGRU-BDLSTM model for the RUL estimation and obtained the RMSE accuracy on the test set in the operating phase.

4.5. Results and Discussions

4.5.1. Prediction performance

In this section, the obtained prediction results from applying each of the proposed hybrid models to the turbofan engines datasets are presented. The purpose of this paper is to make a thorough comparison of the different DL approaches for RUL predictions. The actual and predicted RUL values during the whole life-time of the two randomly selected engines out of several testing engine units across the four datasets (i.e., FD001-FD004) for both methods are depicted in Figure 7 - 10.

It is worth noting that the RUL prediction results for all engine units over the four sub-datasets are precisely predicted, especially for RUL's estimation at the last cycles of the engine unit is more reliable and closer to the true RUL than at the early cycles. Besides, it can be observed that when the RUL engines are large, the accuracy prediction is noticeably higher conversely to the smaller RUL engines (as shown in Figure 7 (b) engine 47). The reason is that when the engine degradation reaches failure, the fault features increase, and that can be extracted through the proposed methods and obtain better perdition results. The RUL's engine is linearly decreasing with time until the degradation engine samples are available. Moreover, accurate estimation of the late period in the engine life cycle plays a crucial role in enhancing operational reliability and system availability, maintaining workplace safety, and reducing maintenance costs.

According to Figure 11, we can easily observe from the distribution of box plots for experiments that the performance of the proposed hybrid model (CAE with BDGRU-BDLSTM) generally performs well on all four sub-datasets, in particular, FD002 and FD004 that are very complicated and the existing models typically fail to provide accurate prediction results for these sub-datasets. Hence, the CAE with the BDGRU-BDLSTM model achieves a good result on FD001 and FD003, the simplest sub-datasets. Table 2 shows the results of both proposed hybrid models in terms of the values of RMSE and score, where IMP is the improvement of the proposed CAE

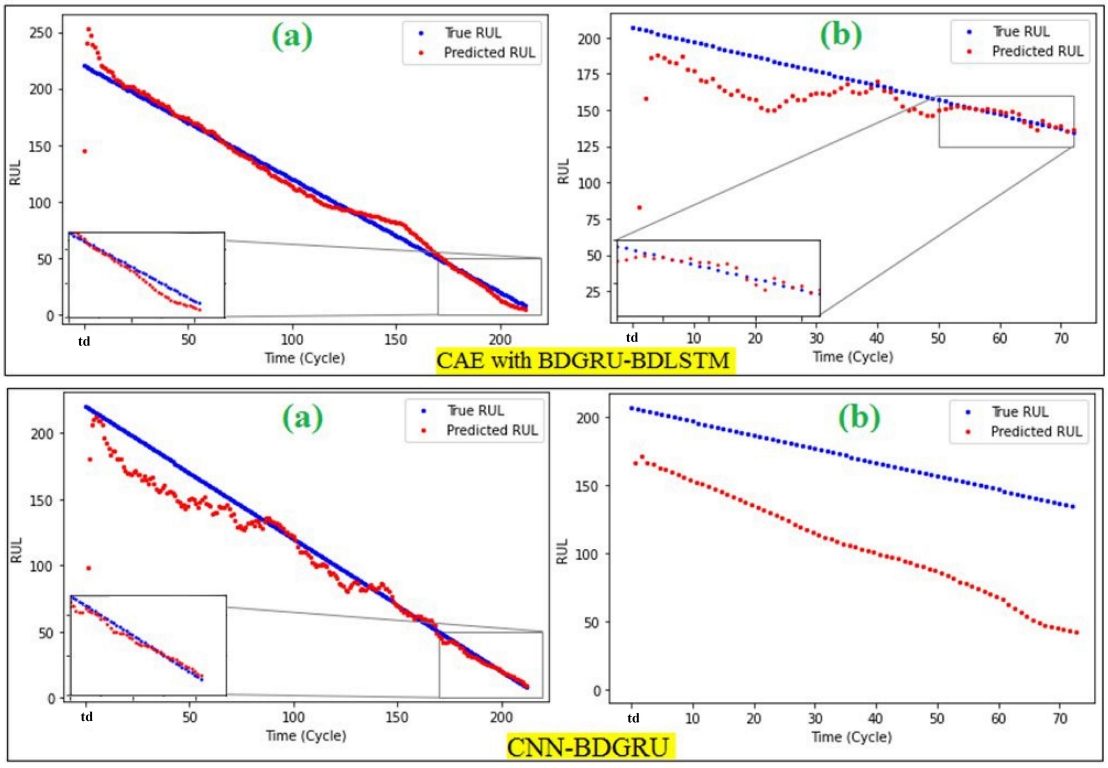

Figure 7. Predicted full life cycles of two testing engine units in FD001 dataset for both hybrid methods: (a) engine #81, and (b) engine #47.

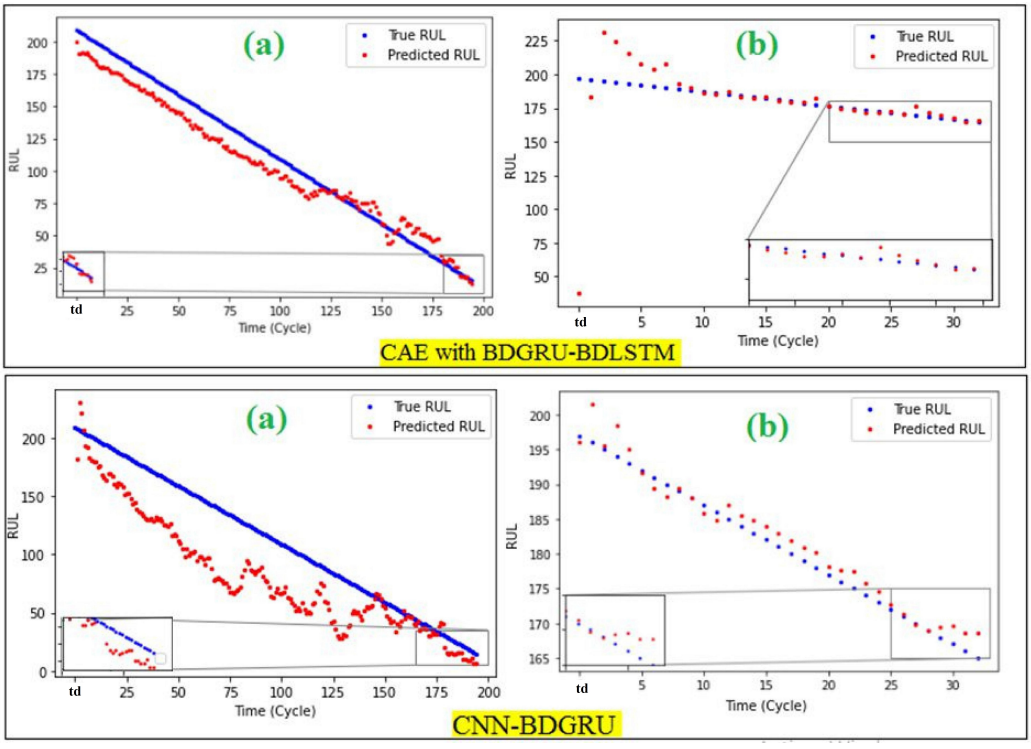

Figure 8. Predicted full life cycles of two testing engine units in FD002 dataset for both hybrid methods: (a) engine #45, and (b) engine #218.

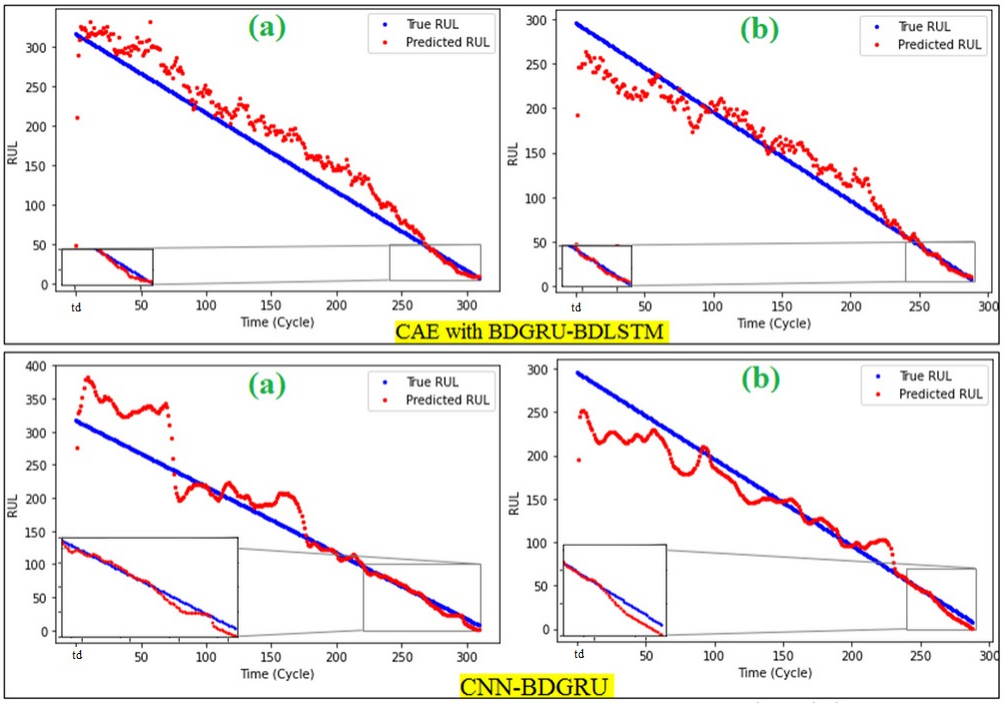

Figure 9. Predicted full life cycles of two testing engine units in FD003 dataset for both hybrid methods: (a) engine #39, and (b) engine #99.

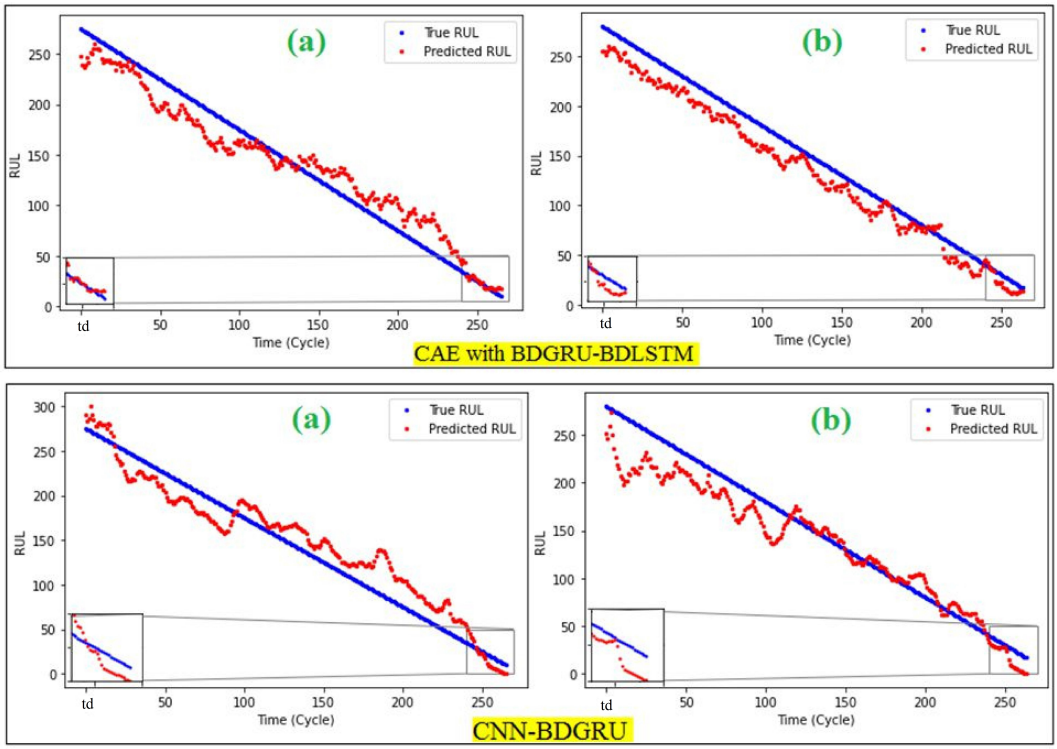

Figure 10. Predicted full life cycles of two testing engine units in FD004 dataset for both hybrid methods: (a) engine #40, and (b) engine #216.

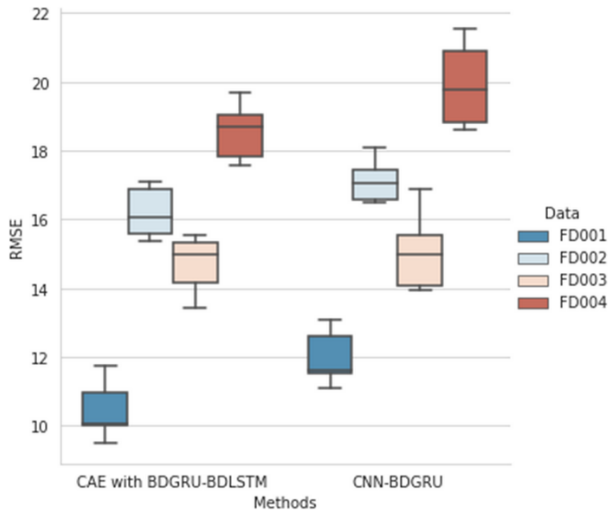

Figure 11. Box plot of the RMSE for both proposed hybrid models over the NASA turbofan engines datasets.

with BDGRU-BDLSTM model over the CNN-BDGRU model. It is defined as IMP= (1-(CAE with BDGRU-BDLSTM / CNN-BDGRU))$\times$100. From the IMP values, we can observe that the CAE with BDGRU-BDLSTM hybrid model consistently obtains RMSE values lower than the CNN-BDGRU model, which has improved the performance in term of RMSE to 14.208%, 6.83%, 3.967%, and 5.537% for group FD001, FD002, FD003 and FD004, respectively.

Table 2. RMSE and score values of the proposed methods on C-MAPSS dataset, where IMP= (1-(CAE with BDGRU-BDLSTM / CNN-BDGRU)) $\times$ 100.

Methods		**FD001**	**FD002**	**FD003**	**FD004**
CAE with BDGRU-BDLSTM	RMSE	9.51	15.35	13.41	17.57
	Score	213	1274	350	1528.18
CNN-BDGRU	RMSE	11.085	16.476	13.964	18.60
	Score	245	1198.42	387	1592
IMP	RMSE	14.208%	6.83%	3.967%	5.537%
	Score	13.06%	-6.3%	9.56%	4.19%

In term of Score values, the proposed hybrid model (CAE with BDGRU-BDLSTM) achieved the lowest Score than the CNN-BDGRU model on FD001, FD003 and FD004, while on FD002, it was a slightly higher Score (worst results). The IMP in term of Score values is around 13.06%, 9.56%, 4.19% for group FD001, FD003 and FD004, respectively.
The error of validation has the same trend of change with training error; both decreases with consecutive epochs tend to be constant in the late period, which proves there is no overfitting problem. It is noted that CAE with BDGRU-BDLSTM hybrid model loss is more stable than CNN-BDGRU hybrid model loss, as depicted in Figure 12.

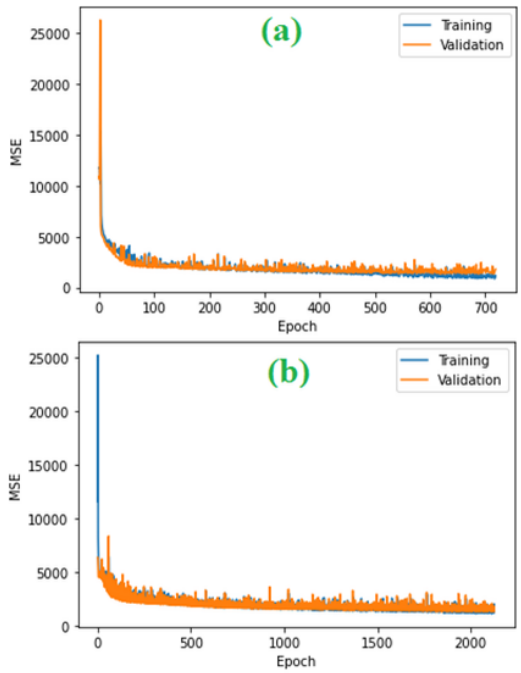

Figure 12. Training and validation loss for FD004 dataset of both hybrid methods: (a) CAE with BDGRU-BDLSTM, and (b) CNN-BDGRU.

For digging into our hybrid models, a feature visualization is a powerful tool. Figures 13 and 14 show the discovered features from the sensor data for both proposed methods (CNN-BDGRU, CAE with BDGRU-BDLSTM) on FD001 subset. Due to the output activation function of BDLSTM and BDGRU is the Tanh function, the values of the features map are in the range of [-1,1], as shown in Figures 13 (b), 14 (b) and 14 (c). As the nodes number of the BDLSTM or BDGRU layer is 50 with both forward and backward layer, it has 100 values of the features map. The activation function of the Convolutional layer in CAE and CNN is the ReLU function, so the output values range in [0, +inf), as shown in Figures 13 (a), 14 (a). Trending patterns are observed along with the time cycles in Figures 13 and 14, which indicate that the proposed methods were able to discover the hidden features.

4.5.2. Computational Cost Analysis

The time complexity for both proposed methods (CAE with BDGRU- BDLSTM, CNN-BDGRU) is discussed in this section. The complexity of the pooling and the Fully Connected layers (FC) takes 5-10 % of the overall computational time (He & Sun, 2015). Therefore, their complexities are not involved in the total time complexity of the proposed models.
The CNN-BDGRU complexity per time step can be calculated as the sum of the complexities of the convolutional lay-

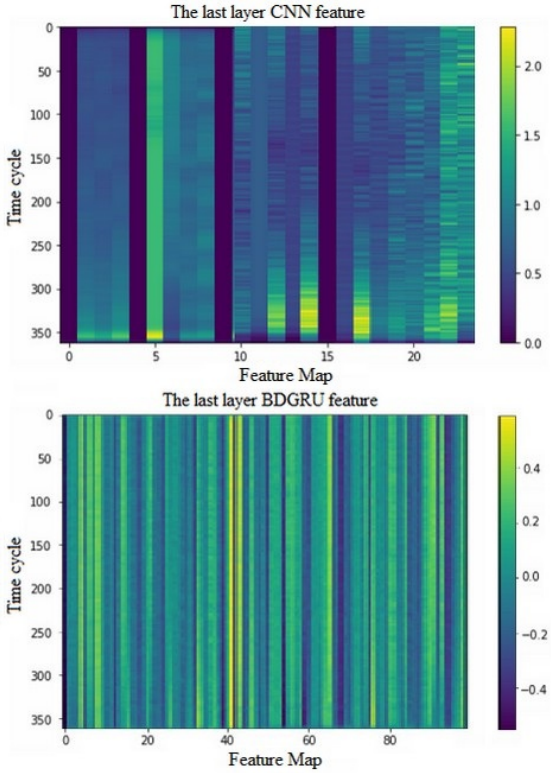

Figure 13. CNN and BDGRU Features visualization of engine 47 of data FD001.

ers and the BDGRU layers (See appendix A.1 for details).

$$O(\sum_{l=1}^{5} n_{l-1}\ s_l^2\ n_l\ m_l^2\ +\ \sum_{i=1}^{2} 2\ W_i) \tag{12}$$

For all the training processes with a function of the input length (x) and epochs (e), the total time complexity is equal to:

$$O([\sum_{l=1}^{5} n_{l-1}\ s_l^2\ n_l\ m_l^2\ +\ \sum_{i=1}^{2} 2\ W_i]\ x\ e) \tag{13}$$

To determine the time complexity of the proposed method that integrates the CAE with BDGRU-BDLSTM, we need to compute the time complexity of convolutional layers, the BDLSTM layers, as well as BDGRU layers. Therefore, its overall time complexity is estimated as Eq.14, as a function of the input length for all the training process.

$$\begin{aligned} &O([(\sum_{l=1}^{10} n_{l-1}\ s_l^2\ n_l\ m_l^2\ )x\ e] + \\ &[(\sum_{i=1}^{2} 2\ W_i\ +\ \sum_{i=1}^{2} 2\ W_i')\ x'\ e']) \end{aligned} \tag{14}$$

We can conclude that the computation time of CNN-BDGRU model is less than the second proposed model (CAE with

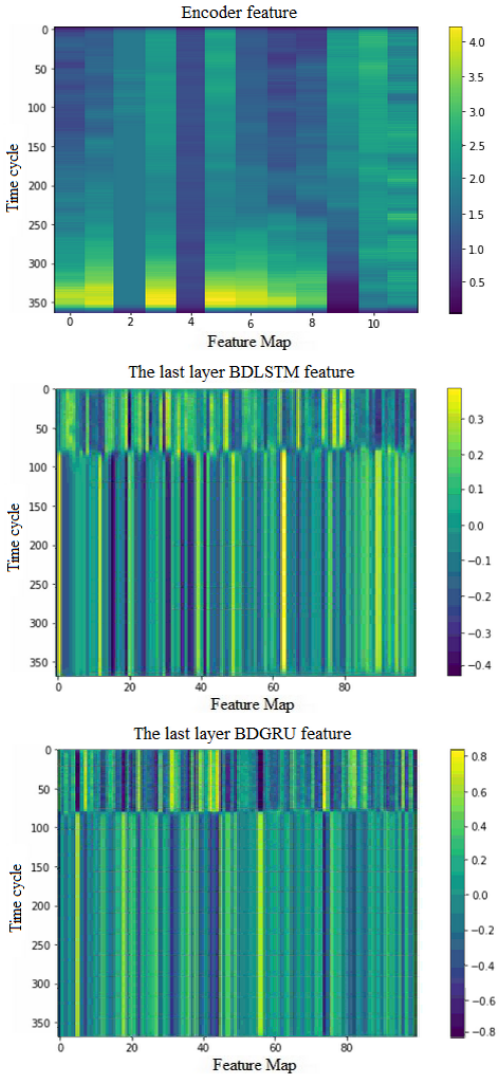

Figure 14. CAE, BDLSTM and BDGRU Features visualization of engine 47 of data FD001.

BDGRU-BDLSTM) in terms of time complexity.

4.5.3. Compared with other approaches

Various prognostic popular methods are performed for comparison purposes, including DNN, RNN, LSTM, GRU, and CNN (as shown in Figure 15). We tried different structures for these methods, and we picked the best ones as follows:

1) **DNN :** contains two hidden layers, which have 50 neurons in each hidden layer and, ultimately, one neuron is attached for RUL estimation.
2) **RNN :** consists of two recurrent layers with hidden units of 50 nodes. Dropout is employed with a rate of 0.25 in each RNN layer.
3) **LSTM :** is implemented with a similar configuration to the RNN method to extract the long-term dependencies.

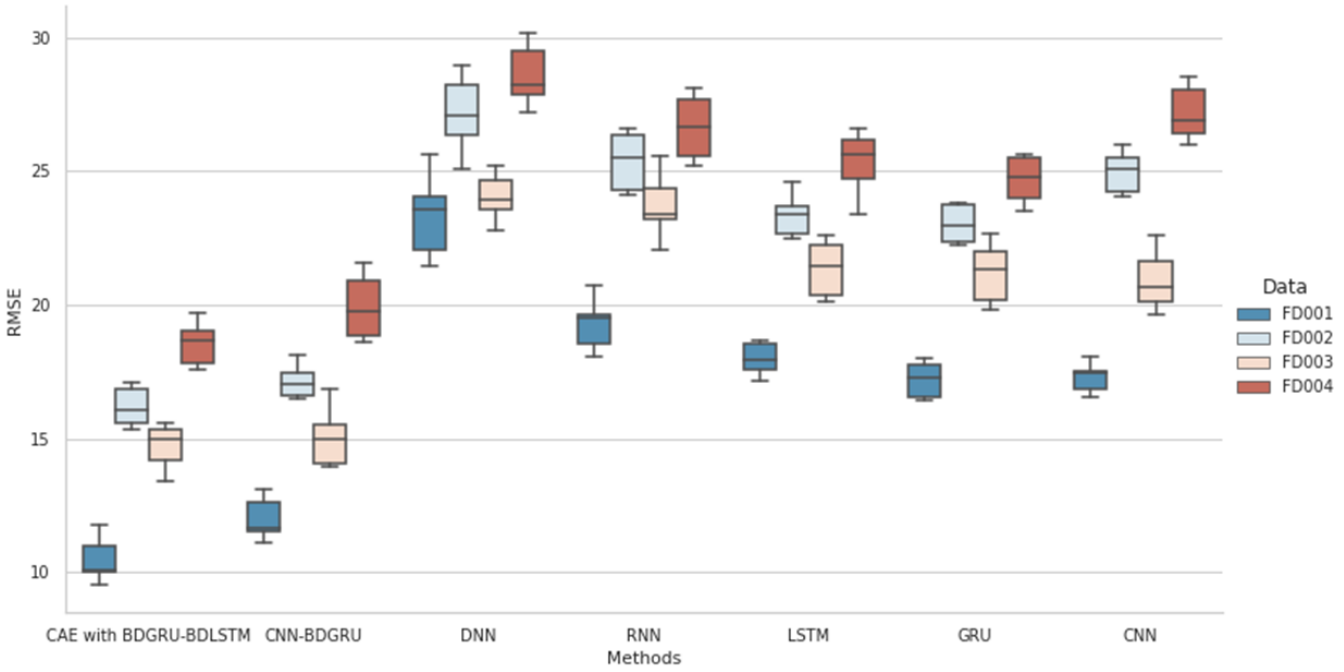

Figure 15. Box plot of the RMSE for both proposed hybrid models compared to the other architectures on the NASA turbofan engines datasets.

4) **GRU :** We use the same configuration of LSTM for the GRU structure for comparison purposes.

5) **CNN :** Five convolution layers are stacked with the same configuration of the CNN path of the proposed CNN-BDGRU method. At the end of this method, one neuron is attached for RUL estimation.

The average performance of DNN, RNN, LSTM, GRU and CNN on each C-MAPSS sub-datasets have been reported in Figure 15 as a RMSE box plot. Among all methods, DNN and RNN performed worse (higher RMSE) than the remaining methods on all four sub-datasets. CNN achieved slightly lower RMSE values than other comparing methods on single operating condition datasets, i.e. FD001 and FD003. On the other hand, GRU and LSTM achieved a lower RMSE than other comparing methods on multiple operation condition datasets, i.e. FD002 and FD004. These results demonstrate powerful of our proposed models, which achieve observable lower average RMSE values (better) in all subsets than other architectures. Besides, the obtained performance from FD002 and FD004 is slightly lower RMSE prediction accuracy, and the reason is that these sub-datasets are more complicated than the FD001 and FD003.

To analyze the results in more detail and to demonstrate the powerful of the proposed CAE as an advanced feature extractor method, the three different features are introduced for comparison purposes. The first kind of features is only raw data with normalization, the second features constructed from the PCA method, and the last features created from the proposed CAE method.

For PCA, the principal components explaining 99% of the data variance were chosen as most appropriate in this study; the original features are reduced to 15 principal components. Considering that $X_p \subset X_m$, where m is the number of original features, and p is the number of principal components, with p<m. The curve of the cumulative sum of variance with the principal components for FD003 using the PCA method is shown in Figure 16.

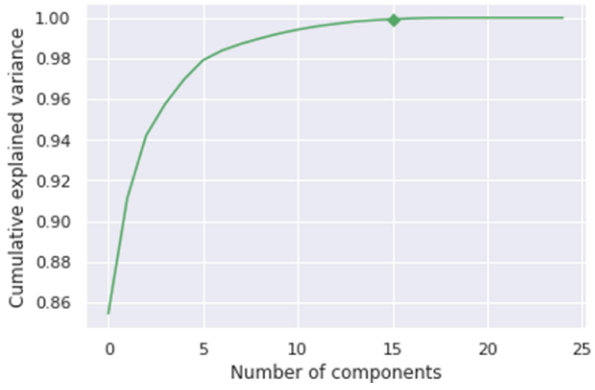

Figure 16. The curve of the cumulative sum of variance with N° of principal components of the C-MAPSS sub-dataset FD003.

Figure 17 shows the distribution box plots of the RMSE testing of multi-model BDGRU-BDLSTM with different features. The proposed BDGRU-BDLSTM multi-model is based on the combination of BDGRU and BDLSTM models simultaneously and in a parallel manner to predict the RUL. "None" indicates that the normalized raw data were used as input. Overall, when trained BDGRU-BDLSTM model on the normalized raw data showed the worst performance over the C-MAPSS datasets. Interestingly, the performance of BDGRU-BDLSTM improved with feature extraction methods. Among the three feature extraction methods, the CAE method can learn robust features than the remaining methods, which gives the best and minimum RMSE values with BDGRU-BDLSTM layers in all sub-datasets.

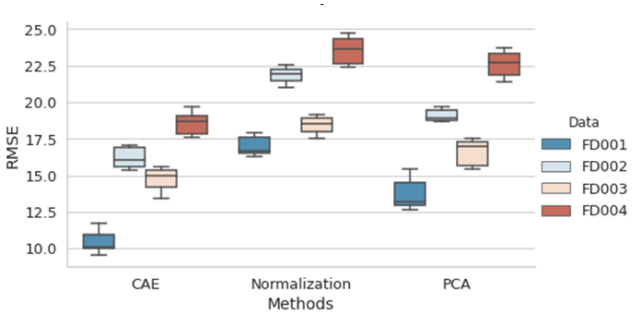

Figure 17. Box plot of the RMSE for BDGRU-BDLSTM method with different features on C-MAPSS dataset.

4.5.4. Effect analysis

To demonstrate the effectiveness of multiple-model DL techniques, we present the effect of combining two DL methods CNN and BDGRU in sequential versus parallel, shown in Figure 18. The comparison result is quantified using RMSE of RUL prediction, and we can notice that the combination of CNN-BDGRU in parallel pathways achieved promising results compared to CNN-BDGRU in a sequential way.

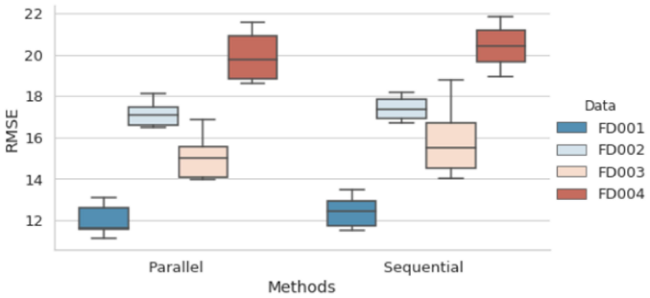

Figure 18. Box plot of the RMSE for CNN-BDGRU in sequential versus parallel.

To verify the validity of the CAE with BDGRU-BDLSTM structure, three experiments are conducted for comparison purposes. In the experiments, we merge CAE once with BDGRU and once with multi-models BDGRU-BDLSTM. According to Figure 19, we can observe that the combination of CAE with the multi-model BDGRU-BDLSTM has achieved good results on all C-MAPSS sub-datasets.

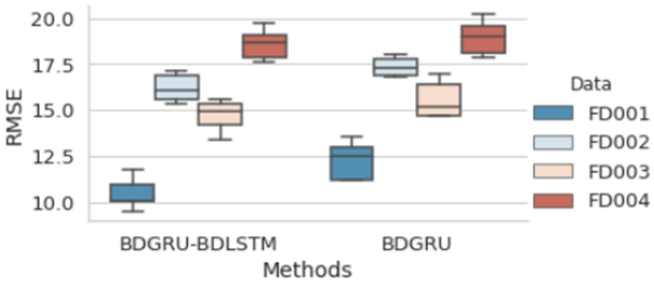

Figure 19. Box plot of the RMSE for CNN-BDGRU in sequential versus parallel.

4.5.5. Comparison with the latest works

Many scholar research has been reported on all sub-datasets C-MAPSS and used in more than 60 publications. Recent studies on the C-MAPSS dataset have been taken into account for comparison to show powerful of the proposed models. Table 3 recapitulates the results of recent studies of the advanced DL methods on the RUL estimation problem extended to all C-MAPSS sub-datasets.

Table 3 shows that the proposed hybrid methods have achieved promising results compared to the recent studies on all C-MAPSS sub-datasets quantified using the RMSE and score metrics. Especially for the complicated datasets FD002 and FD004, the Score and RMSE prediction accuracy obtained from both methods higher than the existing methods. Except on the sub-dataset FD003, the DCNN method (Li et al., 2018), and Deep Bidirectional LSTM (Wang et al., 2018) are slightly higher Score and RMSE (worse) than both our proposed hybrid methods. However, our proposed methods used all three sensor settings and all sensor measurements as input without manually designing features, unlike works (Ma et al., 2018), (Li et al., 2018), and (Wang et al., 2018) that picked 14 sensors data and excluded the three operational settings. Amongst these recent studies, it is worth mentioning that our proposed method that used CAE is the first attempt to adopt CAE in aero-engine prognostic problem in order to extract useful features that serve as inputs for the two separate and parallel pathways (referred to as BDLSTM path and BDGRU path) to obtain more robust features. Furthermore, the reason why our methods are proposed is exhibited superior performance among the most existing methods and for capitalizing on the recent success of multiple-model deep learning techniques and aligning with the power and the success of Convolutional Auto-Encoders to extract automatically useful features with high-level abstractions from com-

Table 3. Performance comparison with the recent DL methods for RUL estimation on the C-MAPSS Dataset.

		CAE with BDGRU-BDLSTM	**CNN-BDGRU**	**CNN (Babu et al., 2016)**	**LSTM (Zheng et al., 2017)**	**CNN (Li et al., 2018)**	**BDLSTM (Wang et al., 2018)**	**BDLSTM (Zhang et al., 2018)**	**AE-BDLSTM (Song et al., 2018)**	**LSTM (Wu et al., 2020)**
FD001	RMSE	**9.51**	**11.085**	18.44	16.14	12.61	13.65	15.42	13.63	18.33
	Score	**213**	**245**	1.2867×10^3	3.38×10^2	273.7	2.95×10^2	/	2.61×10^2	655
FD002	RMSE	**15.35**	**16.476**	30.29	24.49	22.36	23.18	/	/	/
	Score	**1274**	**1198.42**	1.3570×10^4	4.45×10^3	10412	4.13×10^3	/	/	/
FD003	RMSE	13.41	13.964	19.81	16.18	**12.64**	**13.74**	/	/	19.78
	Score	350	387	1.5962×10^3	8.52×10^2	**284.1**	**317**	/	/	828
FD004	RMSE	**17.57**	**18.60**	29.15	28.17	23.31	24.86	/	/	/
	Score	**1528.18**	**1592**	7.8864×10^3	5.55×10^3	12466	5.43×10^3	/	/	/

plex data. We want to point out that the CAE with DBGRU-BDLSTM results are better than our CNN-BDGRU results about 14.208%, 6.83%, 3.967%, and 5.537% in terms of the RMSE values, for FD001, FD002, FD003, and FD004, respectively. However, the time cost of the CAE with BDGRU-BDLSTM is higher than the CNN-BDGRU according to the time complexity metric.

5. CONCLUSION

To leverage the power of various methods, the idea of hybrid methods emerged, ultimately enhancing and obtaining a more accurate prediction. Firstly, we proposed a CAE with a temporal modeling tool that combines BDGRU and BDLSTM models in a parallel manner for degradation features extraction and RUL prediction. We found that the CAE is more suited for data extraction and reduction rather than conventional approaches. Secondly, a hybrid architecture consisting concurrently of CNN and BDGRU models is developed and applied to capture local and temporal features for the RUL estimation. The GRU has appeared as an improved LSTM model with few parameters to increase the training stage's efficiency and speed. Besides, for the extraction of bidirectional temporal features and to prevent the long-term dependency problem, we used a BDGRU. The proposed hybrid models' evaluated results indicate significant improvements over their counterparts and the most robust literature results in terms of RMSE on the C-MAPSS public NASA dataset. We pointed out that the CAE with DBGRU-BDLSTM outcomes reliably performs higher for FD001, FD002, FD003, and FD004 than our CNN-BDGRU outcomes, in terms of the RMSE value around 14.208%, 6.83%, 3.967%, and 5.537%.

As future works, we intend to incorporate an automated method that detects the fault time step of each engine to tackle the problem of RUL that is ill-defined in healthy operation. Due to the black-box nature of the proposed DL, we aim to address the lack of interpretability and transparency of the proposed models using attention mechanisms.

ABBREVIATIONS

PHM	Prognostics and Health Management
RUL	Remaining Useful Life
$C-MAPSS$	Commercial Modular Aero-Propulsion System Simulation
DL	Deep Learning
DNN	Deep Neural Network
CNN	Convolutional Neural Network
AE	Auto-Encoders
DBN	Deep Belief Networks
RNN	Recurrent Neural Networks
$BDRNN$	Bi-directional Recurrent Neural Networks
CAE	Convolutional Auto-Encoder
GRU	Gated Recurrent Unit
$LSTM$	Long-Short Term Memory
$BDGRU$	Bi-directional Gated Recurrent Unit
$BDLSTM$	Bi-directional Long-Short Term Memory
PCA	Principal Component Analysis
LDA	Linear Discriminant Analysis
LLE	Locally Linear Embedding
$KPCA$	Kernel PCA
MLP	Multilayer Perceptron
SVM	Support Vector Machine
RVM	Relevance Vector Machine
$FFNN$	Feed-Forward Neural Network
SAE	Sparse AE
LPC	Low-Pressure Compressor
HPC	High-Pressure Compressor
HPT	High-Pressure Turbine
LPT	Low-Pressure Turbine
$ReLU$	Rectified Linear Unit
$tanh$	hyperbolic tangent
$RMSE$	Root Mean Square Error
MSE	Mean Square Error
$BPTT$	Back-Propagation Through Time
$RMSprop$	Root Mean Square propagation
CPU	Central Processing Unit
RAM	Random Access Memory
FC	Fully Connected

REFERENCES

Atamuradov, V., Medjaher, K., Dersin, P., Lamoureux, B., & Zerhouni, N. (2017). Prognostics and health management for maintenance practitioners-review, implementation and tools evaluation. *International Journal of Prognostics and Health Management*, *8*(060), 1–31.

Babu, G. S., Zhao, P., & Li, X.-L. (2016). Deep convolutional neural network based regression approach for estimation of remaining useful life. In *International conference on database systems for advanced applications* (pp. 214–228).

Belaala, A., Bourezane, Y., Terrissa, L. S., Al Masry, Z., & Zerhouni, N. (2020). *Skin cancer and deep learning for dermoscopic images classification: A pilot study.* American Society of Clinical Oncology.

Chen, J., Jing, H., Chang, Y., & Liu, Q. (2019). Gated recurrent unit based recurrent neural network for remaining useful life prediction of nonlinear deterioration process. *Reliability Engineering & System Safety*, *185*, 372–382.

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on machine learning* (pp. 160–167).

Demšar, U., Harris, P., Brunsdon, C., Fotheringham, A. S., & McLoone, S. (2013). Principal component analysis on spatial data: an overview. *Annals of the Association of American Geographers*, *103*(1), 106–128.

Ding, S.-H., & Kamaruddin, S. (2015). Maintenance policy optimization—literature review and directions. *The International Journal of Advanced Manufacturing Technology*, *76*(5-8), 1263–1283.

Gouriveau, R., Medjaher, K., Ramasso, E., & Zerhouni, N. (2013). Phm-prognostics and health management-de la surveillance au pronostic de défaillances de systèmes complexes. *Techniques de l'Ingénieur*, *9*.

Gouriveau, R., Medjaher, K., & Zerhouni, N. (2017). *Du concept de phm à la maintenance prédictive 1: Surveillance et pronostic* (Vol. 3). ISTE Group.

Guo, C. (2015). Study on the recognition of aero-engine blade-casing rubbing fault based on the casing vibration acceleration. *Measurement*, *65*, 71–80.

He, K., & Sun, J. (2015). Convolutional neural networks at constrained time cost. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 5353–5360).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hsu, C.-S., & Jiang, J.-R. (2018). Remaining useful life estimation using long short-term memory deep learning. In *2018 ieee international conference on applied system invention (icasi)* (pp. 58–61).

Javed, K., Gouriveau, R., & Zerhouni, N. (2017). State of the art and taxonomy of prognostics approaches, trends of prognostics applications and open issues towards maturity at different technology readiness levels. *Mechanical Systems and Signal Processing*, *94*, 214–236.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

Li, X., Ding, Q., & Sun, J.-Q. (2018). Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety*, *172*, 1–11.

Liao, Y., Zhang, L., & Liu, C. (2018). Uncertainty prediction of remaining useful life using long short-term memory network based on bootstrap method. In *2018 ieee international conference on prognostics and health management (icphm)* (pp. 1–8).

Ma, J., Su, H., Zhao, W.-l., & Liu, B. (2018). Predicting the remaining useful life of an aircraft engine using a stacked sparse autoencoder with multilayer self-learning. *Complexity*, *2018*.

Mierswa, I., & Morik, K. (2005). Automatic feature extraction for classifying audio data. *Machine learning*, *58*(2-3), 127–149.

Patro, S., & Sahu, K. K. (2015). Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *science*, *290*(5500), 2323–2326.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Saxena, A., & Goebel, K. (2008). Turbofan engine degradation simulation data set. *NASA Ames Prognostics Data Repository*.

Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management* (pp. 1–9).

Sharma, A., & Paliwal, K. K. (2015). Linear discriminant analysis for the small sample size problem: an overview. *International Journal of Machine Learning and Cybernetics*, *6*(3), 443–454.

Song, Y., Shi, G., Chen, L., Huang, X., & Xia, T. (2018). Remaining useful life prediction of turbofan engine using hybrid model based on autoencoder and bidirectional long short-term memory. *Journal of Shanghai Jiaotong University (Science)*, *23*(1), 85–94.

Tenenbaum, J. B., De Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *science*, *290*(5500), 2319–2323.

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, *2018*.

Wang, J., Wen, G., Yang, S., & Liu, Y. (2018). Remain-

ing useful life estimation in prognostics using deep bidirectional lstm neural network. In *2018 prognostics and system health management conference (phm-chongqing)* (pp. 1037–1042).

Wu, J., Hu, K., Cheng, Y., Zhu, H., Shao, X., & Wang, Y. (2020). Data-driven remaining useful life prediction via multiple sensor signals and deep long short-term memory neural network. *ISA transactions*, *97*, 241–250.

Yuan, M., Wu, Y., & Lin, L. (2016). Fault diagnosis and remaining useful life estimation of aero engine using lstm neural network. In *2016 ieee international conference on aircraft utility systems (aus)* (pp. 135–140).

Zhang, J., Wang, P., Yan, R., & Gao, R. X. (2018). Long short-term memory for machine remaining life prediction. *Journal of manufacturing systems*, *48*, 78–86.

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2019). Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, *115*, 213–237.

Zheng, S., Ristovski, K., Farahat, A., & Gupta, C. (2017). Long short-term memory network for remaining useful life estimation. In *2017 ieee international conference on prognostics and health management (icphm)* (pp. 88–95).

BIOGRAPHIES

Ikram Remadna received the Licence and Master's degree in Computer Science from the University of Biskra, Algeria, in 2014 and 2016. She is now a PhD student in artificial intelligence at LINFI Laboratory University of Biskra, and her current research interest includes Prognostics and Health Management and Deep Learning.

Labib Sadek Terrissa is Professor in computer science at Biskra University, Algeria. He conducts his research activities within LINFI Laboratory where he is the "CoViBio" team head. He is also a senior consultant in digitalization project management. After receiving an engineering degree in electronics, he received his Ph.D. in computer engineering in 2006 from Le Havre University, France. His current research interests include Cloud computing, Machine learning, Smart maintenance, and Prognostic and Health Management.

Soheyb Ayad is an associate professor at the Computer Science Department in the University of Biskra (Algeria). He is also member in LINFI laboratory. He is interested in Networking, Ad hoc Networks, Wire-less Sensor Networks, Cloud Computing, Cloud robotics, Web services, Semantic Web, Internet of Things and Predictive Maintenance fields. He is author of some international publications.

Noureddine Zerhouni holds a doctorate in Automatic-Productivity from the National Polytechnic Institute of Grenoble(INPG), France, in 1991. He was a lecturer at the National School of Engineers(ENI, UTBM) in Belfort. Since 1999, he is Professor of Universities at the National School of Mechanics and Microtechnics(ENSMM) in Besançon. He is doing his research in the Automatic department of the FEMTO-ST Institute in Besançon. His areas of research are related to the monitoring and maintenance of production systems. He is also an expert in adult education in the areas of process improvement and project management.

APPENDIX

A.1 Computational Complexity

The complexity of the CNN layers is calculated as:

$$\sum_{i=1}^{d} n_{i-1} \; s_i^2 \; n_i \; m_i^2$$

where i is the index of a convolutional layer, d is the number of convolutional layers, n_i is the number of filters in the l-th layer, n_{i-1} is the number of input channels of the l-th layer, s_i is the spatial size of the filter, and m_i is the spatial size of the output feature map.

Considering that LSTM is local in both space and time, which means that for each time step LSTM's storage complexity does not depend on the input sequence length (Hochreiter & Schmidhuber, 1997). We conclude that LSTM's complexity per time step and weight is estimated just as O(1). Therefore, the overall complexity of all LSTM layers per time step is equal to:

$$\sum_{i=1}^{d} W_i$$

where W is the number of weights, i is the index of a LSTM layer, d is the number of LSTM layers.

The time complexity of GRU and FC is similar to an LSTM. While the BDLSTM or BDGRU's runtime complexity is increased by twice.

$$\sum_{i=1}^{d} 2 \; W_i$$

Where : For LSTM or GRU: W= KH + KCS + HI + CSI

For FC : W= IH + HK

Where I is the number of inputs units, K is the number of outputs, H is the number of hidden units, C is the number of memory cell blocks, S is the size of the memory cell blocks.