



A Hybrid Deep Learning Framework for Intelligent Predictive Maintenance of Cyber-physical Systems

MAXIM SHCHERBAKOV and CUONG SAI, Volgograd State Technical University

The proliferation of cyber-physical systems (CPSs) and the advancement of the Internet of Things (IoT) technologies have led to explosive digitization of the industrial sector. It offers promising perspectives for high reliability, availability, maintainability, and safety production process, but also makes the systems more complex and challenging for health assessment. To deal with these challenges, one needs to develop a robust approach to monitor and assess the system's health state. In this article, a practical and effective hybrid deep learning multi-task framework integrating the advantages of convolutional neural network (CNN) and long short-term memory (LSTM) neural network to reflect the relatedness of remaining useful life prediction with health status detection process for complex multi-object systems in CPS environment is developed. The CNN is used as a feature extractor to compress condition monitoring data and directly extract significant spatiotemporal features from raw multi-sensory input data. The LSTM is used to capture long-term temporary dependency features. The advantages of the proposed hybrid deep learning framework have been verified on the popular NASA's C-MAPSS dataset. The experimental study compares this approach to the existing methods using the same dataset. The results suggest that the proposed hybrid CNN-LSTM model is superior to existing methods, including traditional machine learning and deep learning-based methods. The proposed framework can provide strong support for the health management and maintenance strategy development of complex multi-object systems.

CCS Concepts: • **Computing methodologies** → **Neural networks**; • **Information systems** → **Decision support systems**;

Additional Key Words and Phrases: Industry 4.0, cyber-physical systems, predictive maintenance, remaining useful life, deep learning

ACM Reference format:

Maxim Shcherbakov and Cuong Sai. 2022. A Hybrid Deep Learning Framework for Intelligent Predictive Maintenance of Cyber-physical Systems. *ACM Trans. Cyber-Phys. Syst.* 6, 2, Article 17 (May 2022), 22 pages. <https://doi.org/10.1145/3486252>

1 INTRODUCTION

With the rapid advancement of Information and Communication Technologies and the integration of advanced analytics into manufacturing, products, and services, many industries are facing new opportunities and at the same time challenges of maintaining their competency and market needs

The reported study was supported by RFBR research projects 19-47-340010 r_a.

Authors' address: M. Shcherbakov and C. Sai, Volgograd State Technical University, Lenin Avenue 28, 400005 Volgograd, Russia; emails: maxim.shcherbakov@vstu.ru, svcuonghvkts@gmail.com.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2022 Association for Computing Machinery.

2378-962X/2022/05-ART17 \$15.00

<https://doi.org/10.1145/3486252>

[16]. Such integration, which is called **Cyber-physical Systems (CPSs)**, combines strengths of optimized industrial manufacturing with internet technologies and changes manufacturing process, maintenance strategies, and maintenance management significantly. In CPSs, the key to realizing automation and intelligence is through big data analysis driven by big computing and big modeling provided by learning techniques such as deep learning [4]. For a summary on the theory and applications of CPSs, we refer the collected edition of Song et al. [40]. CPSs enhances communication between smart manufacturing entities (sensors, actuators, control, etc.) and cyber computational resources to facilitate monitoring, data collection, perception, analysis, and real-time control of manufacturing resources [17]. However, under these conditions, the complexity of production systems necessarily increases as a function of smaller economic margins, faster and safer processes as well as higher demands on product quality and resource efficiency [25]. Currently, a major issue in many industrial settings is how to correlate failure occurrences and the maintenance actions performed to prevent breakdowns [21]. Due to the increasing requirement of reliability, availability, maintainability, and safety of systems, the traditional maintenance strategies (failure-driven and time-based maintenance) are becoming less effective and obsolete [27]. Their main drawback is resource-intensive and high cost, while there is no guarantee of preventing emergency failures.

Due to the growing complexity of CPSs, technical maintenance becomes a challenging issue as the heterogeneity and complexity of systems increase. In this context, intelligent **prognostic and health management (PHM)** technologies that are also known as **predictive maintenance (PdM)**, are showing promising abilities for application and rapidly gaining traction to address the challenge of maintenance of CPSs in the big world of Industry 4.0, where an IoT platform serves as a monitoring and decision-making system [36]. The concept of the PdM has existed for many years, but only recently emerging technologies become both seemingly capable and inexpensive enough to make PdM widely accessible [26]. The PdM can be described as “an intelligent way to maximize the availability of a machine” and refers not only to strategies for early detection and prediction of the condition, but also for taking necessary technical actions appropriate for recognizing and predicting these cases, thereby reducing the maintenance costs, improving the operational performance, and supporting the decision maker [8]. The prediction of **remaining useful life (RUL)** is the core of the PdM. Accurate and reliable RUL assessment result provides decision-makers valuable information to take suitable maintenance strategy to maximize the equipment usage and avoid costly failure, and thus significantly reduce maintenance costs.

Current prognostic approaches can be classified into two main categories: model-based and data-driven approaches [18, 27, 42]. Model-based approaches typically involve building mathematical models to describe the physics of the system states and failure modes; they incorporate a physical understanding of the system into the estimation of system state and/or RUL [22]. Model-based approaches, however, may not be suitable for many industrial applications where the physical parameters and fault modes may vary under different operation conditions [32]. One on hand, it is usually difficult to tune the derived models *in situ* to accommodate time-varying system dynamics. On the other hand, model-based approaches cannot be used for complex systems whose internal state variables are inaccessible (or hard) to direct measurement using general sensors [22].

The data-driven approaches have been proposed and are currently being actively developed with the improvement of sensor technology and signal transmission technology. Data-driven approaches, in contrast, use collected data observations to identify and model relationships that can be used for RUL predictions on new data observations. For this purpose, statistical approaches model the conditional distribution of time lapses to failure given the history of condition monitoring data using several approaches such as regression-based models, Gamma process, Wiener process, or Markovian-based models. While these approaches provide useful estimations of RUL and risk quantification of the solutions, they rely heavily on underlying assumptions of

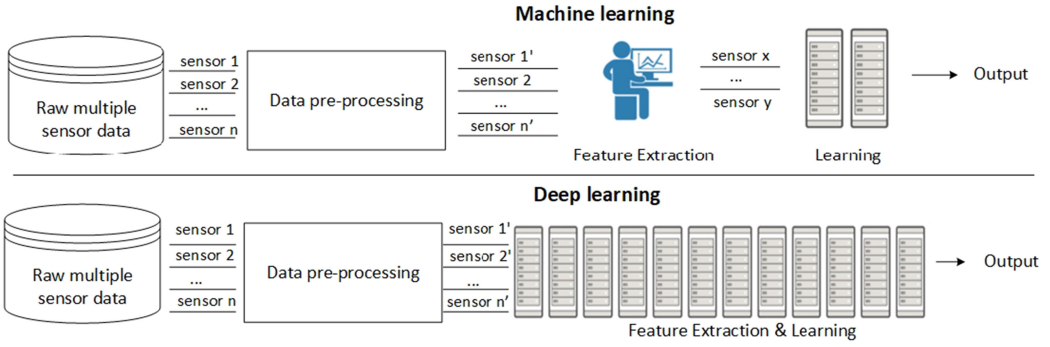


Fig. 1. The difference between machine learning and deep learning in PHM.

distributions or underlying processes from the Wiener or Gamma family. In most cases, these assumptions cannot be verified as fulfilled due to the bias introduced by truncated processes in practice [56].

With the rapid development of advanced sensing technology and affordable storage, it is much easier to acquire mechanical condition data, enabling large scale collection of time series data [33]. With the massive monitoring data, data-driven approaches based on **machine learning (ML)** algorithms has become the most effective solution to address smart manufacturing and industrial big data, especially for performing health perception (e.g., fault diagnosis and remaining life assessment). This approach has the properties of universality, since they are abstracted from the physical nature of the system and do not require deep knowledge of its internal structure and functional relationships between elements. In addition, models based on ML are quite flexible in terms of updating in real-time. In the past few years, various data-driven methods based on ML such as support vector machine [23, 28], random forest, and gradient boosting technique [30] have been proposed for predicting the RUL of degrading equipment and have shown good performance in reliability prognostics. Nevertheless, higher prediction demands make it unfeasible for those traditional data-driven methods to handle a growing number of complicated data [15]. The traditional ML methods are largely dependent on signal processing and feature extraction techniques, which depend on prior domain knowledge and expert experience. As a result, these methods present low efficiency and poor generalization performance [19]. In addition, in these methods, the sequential relationship of multi-temporal observations was not explicitly considered, which might ignore some useful information in time series inputs.

At the same time, as the latest achievement of the machine learning era, **deep learning (DL)** methods provide a powerful solution to the above weaknesses, which has emerged as a potent area to process highly non-linear and varying sequential data with minimal human input within the PHM domain [53]. Unlike traditional ML models that are mostly based on handcrafted features, **deep neural network (DNN)** can operate directly on raw data and learn features from a low level to a higher level to represent the distributed characteristics of data [3, 17], which does not require additional domain knowledge (illustrated in Figure 1). In addition, when processing large amounts of data, the DNN handles the selection of features much better than a human.

More and more DL techniques are developed in field of machine health monitoring. DL techniques, such as **Deep Belief Network (DBN)**, **Convolutional Neural Network (CNN)**, **Recurrent Neural Networks (RNN)**, and their variants, outperformed traditional prognosis algorithms in predictive maintenance for complex multi-object systems [2, 6, 20, 46, 49, 55]. CNNs are suggested for fault diagnosis over multi-channel data from sensors with excellent performance and

lower computational cost, requiring homogeneity of the multi-channel data [1, 2, 9]. In Reference [2], Babu et al. first applied CNN as a regression approach for RUL estimation with multi-sensor raw data as model input. Concurrently, for sequential data such as time-series data, RNNs are considered as more suitable than CNNs [34, 51]. The back-propagation through time algorithm [43] is an adaptation of traditional back-propagation for temporal data used to propagate network's error to previous time instances. However, this propagation can result into vanishing or exploding gradient problem [10], making these networks forget long-term relations. To solve this problem, specific RNN architectures were created based on forget gates, like **long-short term memory (LSTM)** [11]. Zhang et al. [52] applied a method based on LSTM, which is specifically used to discover the underlying patterns embedded in the time series to track the system performance degradation, thereby predicting RUL. In References [46, 49], LSTM was adopted for modeling complex temporal dependencies of multi-sensor raw data for RUL estimation. In Reference [50], Zhang et al. propose a transfer learning algorithm based on **Bidirectional Long Short-Term Memory (BLSTM)** recurrent neural networks for RUL estimation. In Reference [48], both spatial and temporal dependencies are handled by LSTM to identify rotating machinery fault based on the measurement signals from multiple sensors. In Reference [54], Zhao et al. provide an end-to-end framework based on batch-normalization-based LSTM to learn the representation of raw input data and classifier simultaneously without taking the conventional "feature + classifier" strategy. Wu et al. [47] employ SVM to detect the starting time of degradation and used LSTM to predict RUL. In Reference [45], the features are first selected out using correlation metric and monotonicity metric and then fed into LSTM networks with a concatenated one-hot vector.

Although the LSTM and CNN can directly work on raw time series data, both these networks have their drawbacks when processing multi-sensor time-series data containing crucial temporal and spatial dependencies. The time-series data is treated as static spatial data in CNN, where the sequential and temporal dependency are not taken into account, and it may lead to the loss of most information between timesteps [44]. Due to the fact that multi-sensor time-series data of mechanical equipment usually have high sampling rate, the input sequence may contain thousands of timestamps. The high dimensionality of input data will increase the model size and make the model hard to train [33].

In addition, in all the aforementioned studies, the prognostics is treated as a regression problem and as an independent process even though potential relevance exists with the health status detection process. It only provides a predicted RUL value, whose accuracy strictly depends on the prediction horizon, i.e., the period starting from the current time (prediction instant) to real system failure time. Therefore, using the predicted RUL value at the first stage of the system lifetime can lead to a wrong decision. However, this evaluation approach could hide the true overall prognostics accuracy, as the prognostics horizon of the algorithm is not considered.

To address all of the above problems, a new hybrid deep learning framework based on a combination of LSTM and CNN networks for health condition identification and RUL estimation of complex multi-object systems is proposed in this article that is flexible, universal, and reliable enough to show accurate results and be widely implemented for various complex industrial systems. RUL estimation is performed by a developed hybrid CNN-LSTM regression network. Health condition identification is performed by a modified CNN-LSTM network to solve the classification problem. If the current state can be specified, operable and committable time could be measured, and it can be helpful to evaluate the remaining time of a system from the current state [31]. Thus, the proposed framework is suggested to infer the RUL of a current system, reflecting its correspondence to present health conditions. For this purpose, raw sensor measurements with normalization are directly used as model inputs to the network classifier. Subsequently, if system degradation is detected, then the number of RUL is predicted using the

regression model. In this case, no prior expertise on prognostics and signal processing is required, which facilitates the industrial application of the proposed method.

The major contributions of this article can be summarized as follows:

- (1) New data-driven prognostic framework based on a combination of LSTM and CNN networks for effective failure forecasting of complex multi-object systems.
- (2) The proposed hybrid CNN-LSTM networks directly work on raw multi-sensor data with simple pre-processing and can automatically extract significant features without any hand-crafted features or expert experience for technical condition identification and RUL estimation of complex multi-object systems. It can learn both the complex temporal dependency and spatial dependency of multi-sensor time series.
- (3) Verification of the proposed methodology performance through a real application. The proposed hybrid CNN-LSTM models achieve high performance in both classification and regression tasks.
- (4) We achieve higher performance in accuracy compared with the different methods in various metrics.

The rest of this article is organized as follows: The proposed effective data-driven framework based on a combination of CNN and LSTM networks for health condition identification and RUL prediction of complex multi-object systems is presented in detail in Section 2. Section 3 illustrates the experiments performed on benchmark C-MAPSS dataset [35]. By comparing the proposed approach with other existing methods, we demonstrate the high performance of the proposed approach. Section 4 concludes this article and provides future research directions.

2 PROPOSED DATA-DRIVEN PROGNOSTIC MULTI-TASK FRAMEWORK

This section is devoted to introducing the proposed data-driven multi-task framework including two hybrid deep learning network models to reflect the relatedness of remaining useful life prediction with health status detection process for complex multi-object systems in CPS environment is developed. The first model is to identify the stage of degradation of systems. The other model is to estimate the RUL of systems. The proposed framework captures the interdependencies of both tasks offering supportive knowledge to the RUL estimation task to obtain improved prognostic performance.

2.1 Problem Description

We consider a scenario where sensor readings over the operational life of one or multiple instances of a system or a component are available. We denote the set of instances by D . The goal is to predict when a failure of a currently operational instance of the system for which multi-sensor data is available till current time-instance occurs.

More formally, we consider a set of train instances D of a system. Assume that there are n sensors of the same type on each instance. For an instance $i \in D, i = 1, \dots, m$, we consider a multi-sensor time series of sensor readings in a form $X^{(i)} = [x_1^{(i)}, x_2^{(i)}, \dots, x_{T^{(i)}}^{(i)}]$, where $T^{(i)}$ is the length of the time series, $x_t^{(i)} \in R^n$ is an n -dimensional vector corresponding to readings for the n sensors at time t . For a failed instance i , the length $T^{(i)}$ corresponds to the total operational life (from start to end of life) while for a currently operating instance the length $T^{(i)}$ corresponds to the elapsed operational life till the latest available sensor reading.

For a new instance with sensor readings X^* , the goal is to predict when the failure occurs, given the time-series data for train instances $X^{(i)}, i \in D$. Next, we describe our approach based on CNN and LSTM networks that can be easily applied to any complex multi-object system.

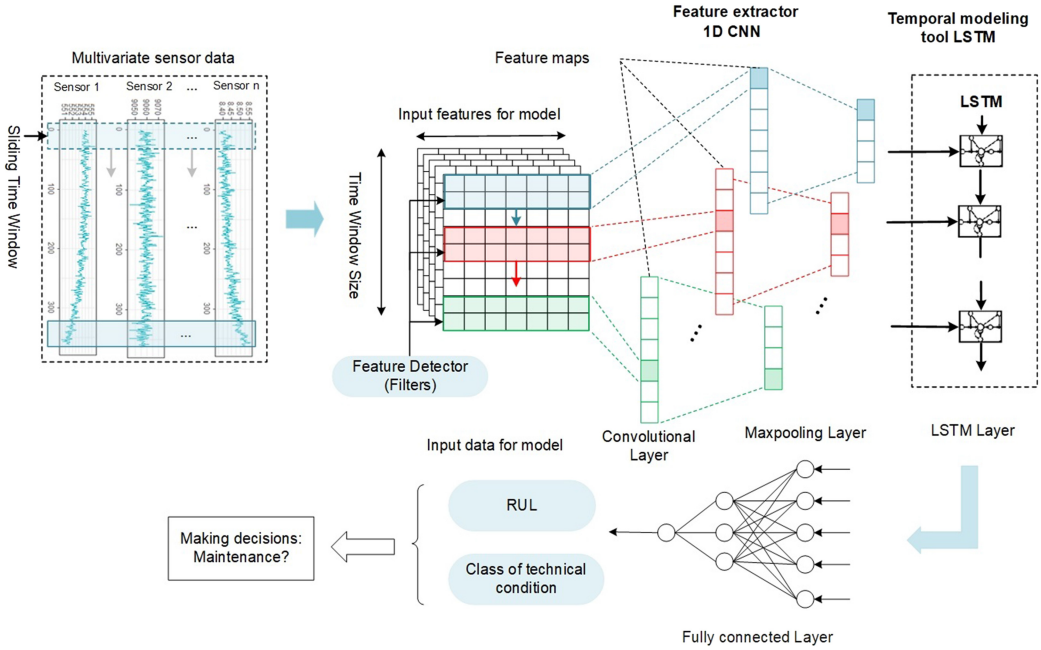


Fig. 2. Proposed hybrid CNN-LSTM networks for intelligent MHM.

2.2 Proposed Hybrid CNN-LSTM Multi-task Framework

Given the complementary strengths of CNN and LSTM networks, in this study, a novel multi-task framework named CNN-LSTM is proposed to effectively predict the RUL by simultaneously learning two tasks: RUL estimation and health condition prediction. To simultaneously estimate the RUL value and present health condition, a hybrid CNN-LSTM deep model is proposed that consists of three substructures, i.e., multiple convolution neural networks, LSTM recurrent neural networks, and fully connected layer for regression. The proposed model captures the interdependencies of both tasks offering supportive knowledge to the RUL estimation task to obtain improved prognostic performance.

Failure prognostics is essentially a time series prediction problem. In the end-to-end model, when the whole sensor signal data is used as input, the difficulty of LSTM training will increase a lot. CNN is introduced to avoid the trouble, which can reduce the original data dimension and extract the significant features. At first, the run-to-failure data is sampled once in a while and it will be divided into some subsequences. Then, every subsequence is set as input to the CNN part for feature extraction. The extracted features from CNN will be packaged into LSTM. The model is shown in Figure 2.

1D CNN for feature extractor. CNN is now the most effective framework with a very strong capacity to discover knowledge behind large data especially for image-based data, since vision is highly hierarchically organized. Good results of CNN for image recognition allow them to be used for processing time sequences, since time can be considered as a spatial dimension, similar to the height or width of a two-dimensional image, but in this case there is a fundamental difference. In the image recognition task, the model accepts a two-dimensional input representing an image's pixels and color channels, in a process called feature learning. For predictive maintenance, considering the collected machinery features are from different sensors in this prognostic problem,

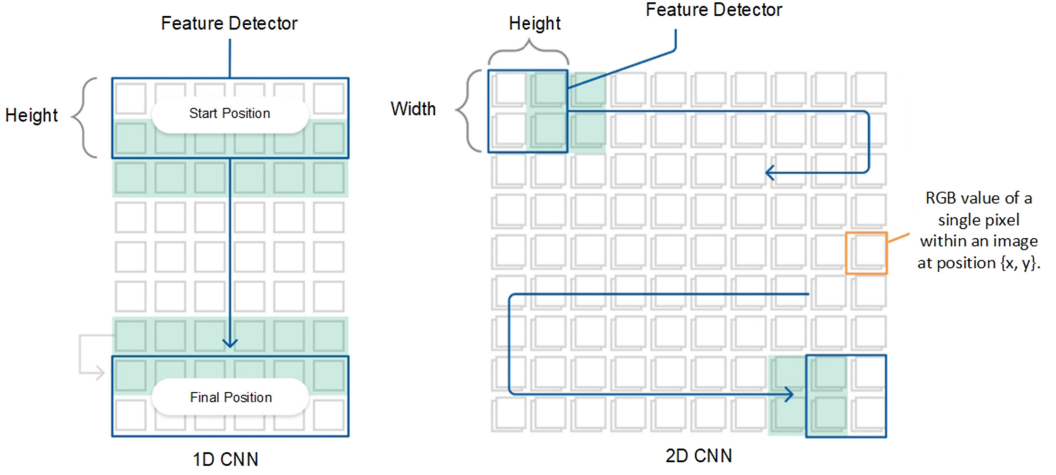


Fig. 3. The difference between 1D CNN and 2D CNN.

the relationship between the spatially neighboring features in the data sample is not remarkable. Therefore, while the input and the corresponding feature maps have 2 dimensions, the convolution filters in the proposed network are **1-dimensional (1D)** to extract features from sequences data and maps the internal features of the sequence. The difference between 1D CNN and 2D CNN is shown in Figure 3.

The input data is the time sequence of multiple sensor measurements $X = [x_1, x_2, \dots, x_{N_L}]$ with the dimension $N_L * N_F$ where N_L denotes the length of the sequence and N_F is the number of selected features. The raw features are usually obtained from multiple sensor measurements. The details of the data preparation will be presented in Section 3.2.

The one-dimensional convolution operation in the convolutional layer can be defined as a multiplication operation between a filter kernel w , $w \in R^{F_L}$, and a concatenation vector representation $x_{i:i+F_L-1}$:

$$x_i = \varphi(w^T x_{i:i+F_L-1} + b), \quad (1)$$

where $*$ denotes the transpose of a matrix $*$, and b and φ represent the bias term and non-linear activation function, respectively.

The output z_i can be considered as the learned feature of the filter kernel w on the corresponding subsequence $x_{i:i+F_L-1}$. By sliding the filter window from the first point to the last point in the sample data, the feature map of the j th filter can be obtained, which is denoted as,

$$z_i = [z_1^j, z_2^j, \dots, z_{N_L-F_L+1}^j]. \quad (2)$$

When using a convolutional neural network, there is a problem—a large number of extracted features, so using such a huge number of features for forecasting was ineffective. To do this, each feature map is compressed to highlight the most significant features by applying a subsampling layer (also known as pooling layer). Average pooling and max pooling are the most common pooling methods. In this article, the max pooling method is adopted:

$$z^{(l+1)} = \text{down}(z^l), \quad (3)$$

where $\text{down}(\cdot)$ represents the subsample function concerning max pooling.

LSTM for modeling temporal dependences. LSTM is a kind of RNN, which is a type of neural network for processing sequence data. RNN contains loops, allowing information to be

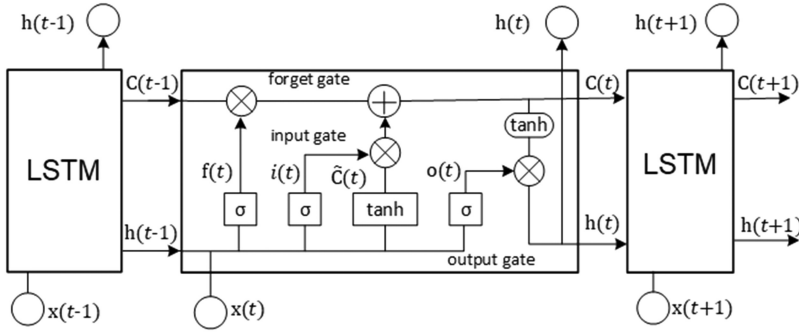


Fig. 4. The structure of three gates in the LSTM cell.

persisted. It can be used to connect previous information to the current task. However, when the interval between the related information and the current predicted position becomes larger, RNN will lose the ability to learn to connect to such far away information. LSTM is proposed to solve the problem. LSTM has been proven to be the most stable and powerful model to learn long-range temporal dependencies in practical application as compared to standard RNNs or other variants. The structure of the repeating module in the LSTM at time t is shown in Figure 4. The LSTM uses three gate structures to control the status of the memory cell $c(t)$. The three gates have the ability to remove or add information to the cell state.

The output $h(t-1)$ and hidden state $c(t-1)$ of LSTM cell at time $(t-1)$ will serve as input of LSTM at time t , which is given by Algorithm 1, which is known as LSTM forward propagation algorithm. LSTM cell also takes sensor data $x(t)$ as an input. There are three gates that control the information flow within cell: (1) input gate $i(t)$ controls what information will be passed to the memory cell based on previous output $h(t-1)$ and current sensor measurement data $x(t)$, (2) output gate $o(t)$ controls what information will be carried to the next timestep, and (3) forget gate $f(t)$ controls how memory cell will be updated.

ALGORITHM 1: LSTM Forward Propagation Algorithm

Input: Feature matrix $x(t)$ and target matrix $h(t)$.

Initialize: Randomly initialize the weight matrix W and bias vector b .

Forget gate: $f(t) = \sigma(W_{xf}x(t) + W_{hf}h(t-1) + W_{cf}c(t-1) + b_f)$.

Input gate: $i(t) = \sigma(W_{xi}x(t) + W_{hi}h(t-1) + W_{ci}c(t-1) + b_i)$.

Candidate value: $g(t) = \tanh(W_{xg}x(t) + W_{hg}h(t-1) + b_g)$.

Update the cell state: $c(t) = f(t) \otimes c(t-1) + i(t) \otimes g(t)$ where \otimes is the element-wise multiplication.

Update the output of the LSTM:

$$o(t) = \sigma(W_{xo}x(t) + W_{ho}h(t-1) + W_{co}c(t) + b_o)$$

$$\hat{h}(t) = o(t) \otimes \tanh(c(t)).$$

Outputs: The estimation value $\hat{h}(t)$.

The fully connected layer connects all the nodes between the adjacent layers, so the features extracted from the front can be integrated. Due to its fully connected nature, the fully-connected layer usually has the largest amount of parameters. The calculation process of the fully connected layer can be expressed by Equation (4).

$$Y = \varphi(W_{fc}x + b_{fc}), \quad (4)$$

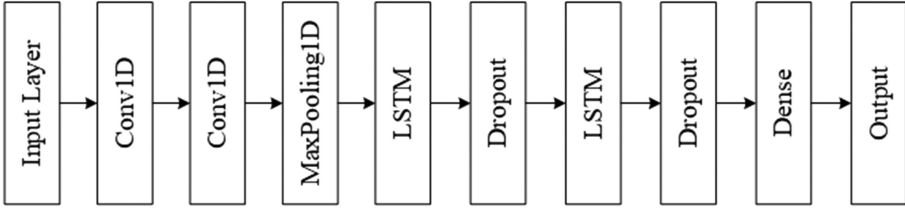


Fig. 5. Architecture for estimating RUL.

where W_{fc} is the weight matrix of the fully connected layer, b_{fc} is the bias vector, x is the input, and Y is the output matrix.

2.3 Proposed CNN-LSTM Network Architecture for Estimating RUL

This subsection aims to describe the architecture and configuration of the proposed hybrid CNN-LSTM regressive network for the system RUL estimation. In this case, the hybrid CNN-LSTM network is implemented for regression and can be called as CNN-LSTM regressor. Figure 5 shows its architecture that has four types of layers: the convolutional, the pooling, the LSTM, and the fully connected (Dense) layers.

To avert over-fitting, which is prone to occur when training deep learning models, two regularization techniques—Dropout and Early stopping—are applied:

- (1) *Dropout* [41] is a recently introduced effective algorithm for training neural networks by randomly dropping units during training to prevent their co-adaptation. This helps the model to produce robust units that do not depend on the details of the activation of other individual units. In this work, we use the same dropout mask at each timestep for each unit of the LSTM.
- (2) *Early stopping* is one of the most popular, and also effective, techniques to prevent over-fitting. In the training process, the 10% training data is selected to compute the loss function at the end of each training epoch, and once the loss stops decreasing, stop the training and use the test data to compute the final model accuracy.

To extract features from the “raw” input signals, we utilize a 1D CNN network that is composed of two convolutional layers with the number of filters F_N and the filter length F_L for each layer and one max pooling layer. The *ReLU* function was used as the activation function for each convolutional layer: $\varphi(x) = \max(0, x)$.

The multiple fault features extracted from the 1D CNN network are added as the input of the next LSTM network to adaptively capture long-term dependencies and nonlinear dynamics of time series data. Two LSTM layers are sequentially stacked in the network. For LSTM configuration, the number of the memory units for every layer have to be provided. It is necessary to specify that the LSTM units return all the outputs from the unrolled LSTM unit through time. Then, it allows learning sequences of observations and it is well adapted to time series problem. However, there is an overfitting on training data. Therefore, the “Dropout” regularization method is used for every LSTM layer to improve the model performance.

Finally, fully connected layer is built on top of CNN-LSTM to process the outputs of LSTM, since it is good at mapping the learned feature representation to sample labels. In the output layer there is 1 neuron that will gives us back the number of RUL and a linear activation.

The model weights are continuously updated by the backpropagation algorithm, as shown in Figure 6.

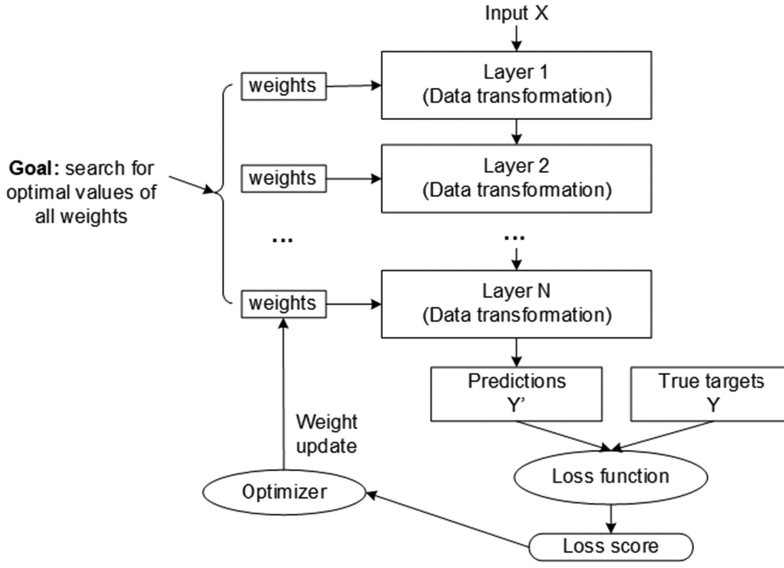


Fig. 6. Loss estimation as feedback for weight adjustment.

We choose **Mean Square Error (MSE)** as the loss function and use RMSProp optimizer [5] to minimize the loss function. In this case, the the network's weights are updated by minimizing the MSE function using the RMSprop optimizer algorithm (see Figure 6). The MSE function is formulated as follows:

$$MSE = \frac{1}{N} \sum_{t=1}^N |RUL_{y_i} - RUL_{\hat{y}_i}|^2, \quad (5)$$

where RUL_{y_i} and $RUL_{\hat{y}_i}$ are the true value and predicted RUL, respectively. N is the total number of samples in the testing set.

2.4 Proposed CNN-LSTM Network Architecture for Identifying Technical Condition

From the start of operation until the limit state is reached, the system will be in different technical conditions (classes) $C = [C_1, C_2, \dots, C_n]$, where n - is the number of class labels. In real decision support systems, there is no practical need to consider more than three classes $C = [C_n, C_w, C_c]$:

- (1) C_n - "Normal" condition of system. Technical maintenance is not required.
- (2) C_w - "Warning" or approaching the pre-failure condition of the system. The necessary planning for technical maintenance and repair.
- (3) C_c - "Critical" condition of the system. Operation is not allowed. Urgent action on the system is required.

In this statement, the problem of identifying the technical condition of complex multi-object systems can be considered as a classification problem with several classes. Note that in a simple case, from the point of view of technical diagnostics, we can consider a simplified problem—the problem of binary classification: According to a given vector of system parameters, it is necessary to determine which condition (healthy or faulty) the system belongs to. Therefore, the binary classification is regarded in this article. To solve this problem, the proposed hybrid CNN-LSTM model for RUL estimation was modified as follows:

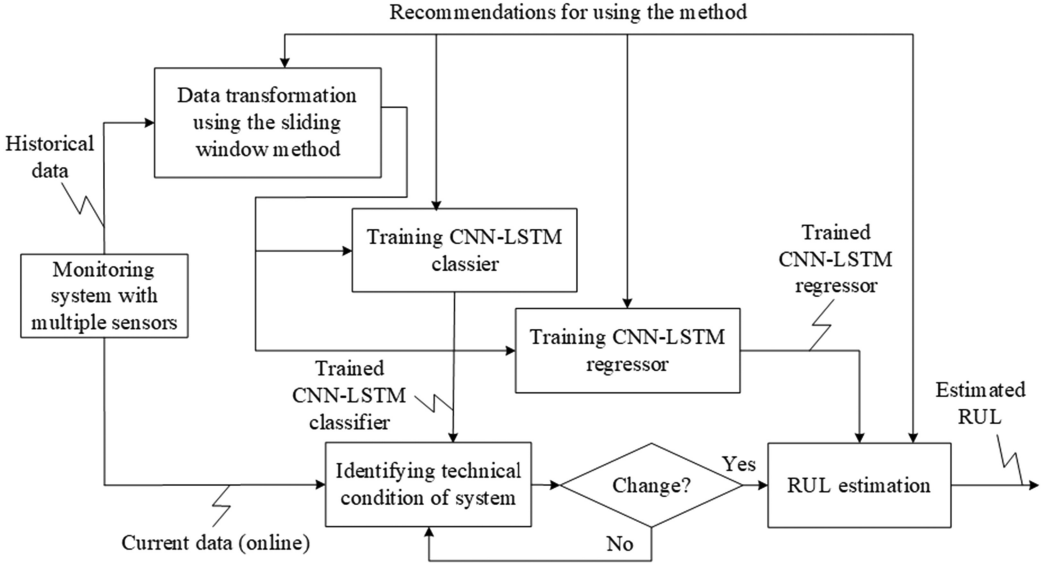


Fig. 7. A final data-driven prognostic scheme.

- In the output fully connected layer of the network, one output with a “sigmoid” activation function is used for the binary classification problem.
- The loss function for the binary classification problem is binary cross-entropy/log loss, which looks like this:

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^N (y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))), \quad (6)$$

where y is the label (0 for healthy condition and 1 for faulty condition) and $p(y)$ is the predicted probability of the condition being healthy for all N samples.

- The Adam optimizer [13] is implemented to train the model. In this case, the weights were updated by minimizing the function $H_p(q)$ using the Adam optimizer algorithm.

2.5 Final Prognostic Scheme

Figure 7 shows the final proposed data-driven prognostic scheme that consists of two trained hybrid CNN-LSTM networks described above. The first one is a classifying network that predicts the class of an engine according to its state. The second network is regressive and allows to accurately calculate the number of RUL.

The classifying network only provides a Boolean answer, but can provide greater accuracy with less data. The regressive network needs more data, although it provides more information about when the failure will happen. However, as noted in Section 1, training a single regressive network to predict the RUL of a machine provides a greater amount of errors at the first stage of the system lifetime, consequently, using the predicted RUL value at the first stage of the system lifetime can lead to a wrong decision. This is why the proposed prognostic scheme has been chosen.

During the training stage, the proposed hybrid network models take the sensor measurement sequences $X_i, i = 1, 2, \dots, m$, to learn to which time window the true RUL and technical condition belongs. Next, during the test stage (the application stage), at time t , the constructed CNN-LSTM classifier will take the vector of sensor measurements x_t as input data and output the technical

condition belongs to determined time window. If the technical condition is defined as no healthy, then the constructed CNN-LSTM regressive model will be used to predict RUL.

3 EXPERIMENTAL STUDY

In this section, the performances of proposed hybrid CNN-LSTM models are empirically evaluated on a well-known benchmark C-MAPSS dataset from NASA. First, descriptions of the C-MAPSS dataset are presented. Then details of the experimental procedure are provided. At last, the proposed models are compared to benchmark models and recent studies in this problem.

3.1 C-MAPSS Dataset

Due to the difficulty of obtaining real data and measurements from companies it has been decided to create a functional prototype of predictive maintenance using a public database. To test the proposed approaches, we used the *Turbofan Engine Degradation dataset* from Aero-Propulsion System Simulation (C-MAPSS), which was developed by the Prognostics Center of Excellence at NASA's Ames research center [35]. It contains sensor readings from a fleet of simulated aircraft gas turbine engines, recorded as multiple multivariate time series. All engines are of the same type, but each engine starts with different degrees of initial wear and variations in the manufacturing process, which is unknown and considered to be healthy. The data includes the following sets:

- *Training set*: It is the aircraft engine's run-to-failure data.
- *Testing set*: It is the aircraft engine's operating data without failure events recorded.
- *Ground truth data*: It contains the information of true remaining cycles for each engine in the testing data

Both the training set and the test set includes 26 columns: engine number, timestep, three operational settings, and 21 sensor measurements. See Reference [35] for a deeper understanding of the C-MAPSS dataset. The purpose is to prognosticate the numbers of remaining operational cycles before failure in the testing set, i.e., the number of operational cycles after the last cycle in which the engine will continue to run normally.

The Figure 8 shows the visualization of sensor readings over time for a random sample of 10 engines from the training set, plotted against time. Note that each subplot contains 10 lines (one for each engine). There are three operational settings (c_1, c_2, c_3) that can be used to change the performance of each machine. Each engine has 21 sensors (s_1, s_2, \dots, s_{21}) collecting different measurements related to the engine state at runtime. Collected data is contaminated with sensor noise. It is apparent from this figure that, perhaps due to their different initial conditions, each engine has a slightly different lifetime and failure pattern. This highlights the fact that each engine's progress in time is not quite aligned with any other.

3.2 Data Pre-processing

Before training the hybrid CNN-LSTM network models, it is necessary to process the heterogeneous data from multiple sensor sources as follows:

- (1) **Data selection**: There are 3 operational settings and engine unit measurements from 21 sensors in the C-MAPSS dataset that can provide degradation information of the engine units. However, it can be observed from Figure 8 that parts of the sensor samples do not show significant variations as the life cycle increases. These data cannot provide useful information about RUL and will increase the complexity of neural networks. Therefore, these features will be removed from the dataset, whose indices are $c_3, s_1, s_5, s_{10}, s_{16}$, and s_{19} .

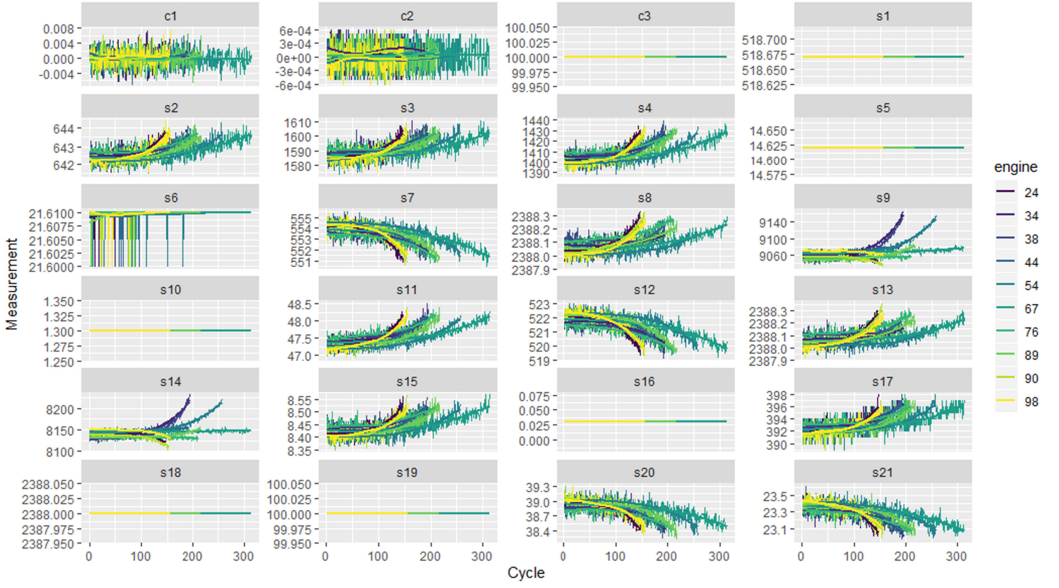


Fig. 8. Sensor readings over time for the first 10 engines in the training set.

- (2) **Data Normalization:** The input measure data from multiple sensor sources is complex and multidimensional with different ranges of values. To use these heterogeneous data for training the proposed network models, it is necessary to normalize every feature value. The collected measurement data from each sensor are normalized to be within the range of $[0, 1]$ using the min-max normalization method:

$$x_i^* = \frac{x_i - x_{min}}{x_{max} - x_{min}}, \quad (7)$$

where x_{max} and x_{min} are the maximum, and minimum with respect to each sensor, respectively.

- (3) **Data labeling:** To perform the regression and classification, it is necessary to label the data. Let us look at the specifics of label assignment for both tasks:

- **Regression task:** For each engine trajectory within the training sets, the last data entry corresponds to the moment the engine is declared unhealthy or failure status, i.e., the length $T^{(i)}$ responds to the total operational life (from start to end of life). So, we can assign target values of RUL for training regressive network as follows:

$$RUL^{(i)}(t) = T^{(i)} - t^{(i)}, T^{(i)} > t^{(i)}, \quad (8)$$

where $T^{(i)}$ is the time of end life of the i th engine and t its current operating time. In this case, RUL decreases linearly along with time. This definition implies that the health of a system degrades linearly along with time. In practical applications, degradation of a component is negligible at the beginning of use and increases when component approaches end-of-life. To better model the RUL changes along with time, in References [2, 7, 20], a piece-wise linear RUL target function was proposed, which limits the maximum RUL to a constant value and then starts linear degradation after a certain degree of usage. In this article, for C-MAPSS datasets, we set the maximum limit as 130 time cycles.

- *Classification task*: For binary classification, the information is required whether the system fails in time window (horizon) w , then the data will be labeled by two classes. The first class, noted class 1, represents the case where the system remaining life time will be greater than the time window, i.e., $RUL > w$ (healthy condition). The second class, noted class 2, concerns the case where the systems remaining useful life will not exceed the time window w , i.e., $RUL \leq w$ (faulty condition).
- (4) **Time-window Processing**: Before training the proposed models, time-window processing is used to generate the network inputs by sliding the window through the raw data. The window length is denoted as N_L and the sliding stride is expressed as k . Considering that short sliding strides can increase the number of training samples, which may reduce the risk of over-fitting, it is reasonable to set $k = 1$. Input pairs can be expressed as $I = [X_{window_start}, \dots, X_{window_end}]$. The input size of CNN-LSTM, as noted above, is $N_L * N_F$. The data samples in a time window belong to the same engine unit. The target class and RUL of the last data point are used as the target label for each time window. The value of the parameter N_L should be smaller than that one of the recorded data length, i.e., the length of the recorded trajectories. In addition, the greater value of N_L is, the higher capacity of looking back in the history data of the models will be. However, this will lead to an increase of the training time. Therefore, for every test set in the case study, it is preferable to have the values of N_L large enough to benefit from the history information, but must be smaller than the minimum length of the recorded trajectories. Note that the minimum recorded length in C-MAPSS FD001 dataset is 31 cycles, therefore, the value of N_L is set to 30 cycles.

3.3 Benchmark Models

In this article, a variety of benchmark models is designed to prove the hybrid CNN-LSTM model's effectiveness. The benchmark models fall into two main categories: DL models and traditional ML models. The DL models include **Multilayer Perceptron (MLP)**, CNN, and LSTM neural networks. The traditional ML models include **Linear Regression (LM)**, **Regression Trees (CART)**, **Support Vector Machines (SVM)**, **Random Forests (RF)**, **Bootstrapped Aggregation (Bagging)**, and **Extreme Gradient Boosting (XGBoost)**.

Even though machine learning demonstrates high predictive power, these models struggle with the nature of sensor data as time series [14]. Unlike DL network models such as CNN and LSTM that can learn from sequences of varying lengths, rather than a fixed size of feature vector, all traditional ML models take only the current sensor measurements X_t when predicting the system failure. However, this approach ignores the trajectory of historic sensor data. Yet the history of sensor measurements is likely to encode valuable information regarding the past deterioration and usage of machinery [14]. As a remedy, the literature suggests using various feature engineering methods as a means of encoding the past usage of machinery into an input feature vector for the predictive model. Yet previous research has only little guidance at hand regarding what type of features are most useful. Hence, in this article, we explore various feature engineering methods that are mentioned in Reference [24] to improve the performance of traditional ML models. To optimize hyperparameters for each traditional ML model, the grid search technique is used.

As DL models, we use the following three neural networks:

- (1) **MLP network**. The MLP is constructed by using four hidden layers. The Relu function is the activation function in each hidden layer.
- (2) **CNN network**. In this article, we adopt CNN consisting of two convolutional layers to extract spatial features and the fully connected neural network to obtain a regression model [2].

- (3) **LSTM network.** We adopt two LSTM layers to extract long-term temporary dependencies and the fully connected neural network to achieve RUL prognostics [55].

To determine the best hyperparameter combination for all DL models, we use keras tuner library [29], which will be presented in Section 3.5.

3.4 Performance Metrics

3.4.1 Performance Metrics for the Regression Task. Measuring forecast accuracy (or error) is not an easy task, as there is no one-size-fits-all measure. Due to the fact that each error measure has the disadvantages that can lead to inaccurate evaluation of the forecasting results (each error measure will avoid some pitfalls but will be prone to others) [37], the choice of an appropriate performance metric depends on the specific case. The time series used in this article have the same scale and the actual values are not equal to zero, so it is recommended to use **Mean Absolute Error (MAE)**, **Root Mean Squared Error (RMSE)**, and **Mean Absolute Percentage Error (MAPE)** that are three of the most common metrics used for evaluating and reporting the performance of a regression model. Moreover, the error measurements mentioned above common-used in the field of RUL prediction. In addition, using only one evaluation metric maybe can not wholly measure your model's performance. Often it is preferable to use multiple metrics and compare results together to get better insight. Evaluation metrics are formulated as follows:

$$MAE = \frac{1}{N} \sum_{i=1}^N |RUL_{y_i} - RUL_{\hat{y}_i}|, \quad (9)$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (RUL_{y_i} - RUL_{\hat{y}_i})^2}, \quad (10)$$

$$MAPE = \frac{1}{N} \sum_{i=1}^N \left| \frac{RUL_{y_i} - RUL_{\hat{y}_i}}{RUL_{y_i}} \right|, \quad (11)$$

where RUL_{y_i} and $RUL_{\hat{y}_i}$ are the prediction value and true RUL, respectively. N is the total number of true RUL targets in the respective testing set.

In addition, using only one evaluation metric maybe can not wholly measure your model's performance; it is often preferable to use multiple metrics and compare results together to get better insight.

3.4.2 Performance Metrics for the Classification Task. The accuracy of a classification model for the classification problems is evaluated using a confusion matrix, which contains information about actual and predicted classifications done by a classification model. A confusion matrix has two dimensions; one dimension is indexed by the actual class of an object, the other is indexed by the class that the classifier predicts. Table 1 presents the basic form of confusion matrix for two-class classification task (in this case, healthy and not healthy or faulty). In such a representation the columns represent the predicted class labels of the data, while the rows represent the actual classes. Assuming (arbitrarily) that the class labels are "0" and "1" and that these are considered "positive" and "negative," respectively. In this setting, the diagonal elements of the confusion matrix, the **true positive (TP)** and **true negative (TN)** rates, are the observations that are correctly classified by the algorithm. The **false negatives (FN)** are positive observations that have been wrongly labelled as negative by the algorithm. Contrarily, the **false positives (FP)** are the negative observations that have been incorrectly classified as positive. Using this notion, a number of metrics regarding the predictive performance of the model can be derived.

Table 1. Confusion Matrix for Two-class Classification Problems

		Predicted class	
		Healthy (0)	Faulty (1)
Actual Class	Healthy (0)	True Positive (TP)	False Negative (FN)
	Faulty (1)	False Positive (FP)	True Negative (TN)

A number of measures of classification performance can be defined based on the confusion matrix. Some common measures are given as follows:

Accuracy is one of the most commonly used measures for the classification performance, and it is defined as the proportion of true results (both true positives and true negatives) in the population. It is calculated as a ratio between the correctly classified samples to the total number of samples as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (12)$$

Precision is a measure of the accuracy provided that a specific class has been predicted. It is defined by:

$$Precision = \frac{TP}{TP + FN}. \quad (13)$$

Recall is a measure of the ability of a prediction model to select instances of a certain class from a dataset, and it is defined by the formula:

$$Recall = \frac{TP}{TP + FP}. \quad (14)$$

The traditional **F-score (F1 score)** is the harmonic mean of precision and recall:

$$Fscore = \frac{2 * Recall * Precision}{Recall + Precision}. \quad (15)$$

3.5 Experimental Results and Discussions

Experiments are conducted on the Google Colaboratory cloud service (also known as Colab) with 12 GB NVIDIA Tesla K80 GPU. Programming language is Python 3.8, and deep neural network models are created using the Keras and Tensorflow libraries. In the training procedure, the input data is divided into several mini-batches. The batch size is 200 and the maximum training epoch is set to be 100.

To determine the best hyperparameter combination in proposed deep learning models, we use keras tuner library [29]. The library search function performs the iteration loop, which evaluates a certain number of hyperparameter combinations. Evaluation is performed by computing the trained model's accuracy on a held-out validation set (see Figure 9). For this purpose, a random search technique has been used with the following hyperparameters:

- Number of filters in each convolution layers: range between 16 and 256 with step = 16
- Kernel (filter) size in each convolution layer: range between 1 and 10 with step = 1
- Numbers of neurons in each LSTM layer: range between 10 and 100 with step = 10
- Dropout rate: range between 0.0 and 0.5 with step = 0.1.

The obtained optimal hyperparameters of the proposed CNN-LSTM models are summarized in Table 2.

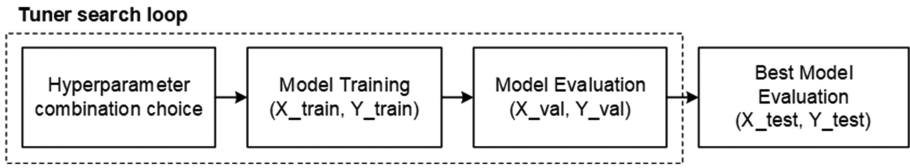


Fig. 9. Hyperparameter tuning process with keras-tuner).

Table 2. Hyperparameter Setting of CNN-LSTM Networks

Layer Index	Type	Filters/ Neurons	Filter Size	Region	Activation function	Dropout rate
1	Conv1D	32	5	-	ReLu	-
2	Conv1D	64	3	-	ReLu	-
3	MaxPooling1D	-	-	3	-	-
4	LSTM	50	-	-	Tanh	0.2
5	LSTM	50	-	-	Tanh	0.2
6	Dense (classification)	1	-	-	sigmoid	-
7	Dense (Regression)	1	-	-	linear	-

Table 3. Experimental Results of Different Prediction Models

Model	MAE	RMSE	MAPE
LM	25.97	31.25	58.17
CART	19.25	24.76	50.56
Bagging	21.37	27.38	47.25
SVM	16.01	21.84	28.16
RF	17.51	23.94	31.37
XGBoost	16.38	22.72	30.18
MLP	17.79	23.02	36.74
CNN	14.15	17.98	28.98
LSTM	11.98	16.06	15.80
The proposed CNN-LSTM	9.86	12.76	13.21

The proposed model has been trained with two dropout layers with a drop rate of 0.2 and early stopping; a comparatively better result is obtained that has the MAE in the training and validation sets of 9.06 and 9.23, respectively. This techniques are considered as a way to avoid overfitting. If proposed model has been trained without using dropout and early stopping, then it gives poor results due to overfitting (MAE for the training and validation sets of 8.67 and 15.69, respectively). The same techniques are used for all neural network models.

3.5.1 *Regressive Network for RUL Prediction.* Table 3 shows the performance metric values (MAE, RMSE, and MAPE) of benchmark models and the proposed hybrid CNN-LSTM model.

We can observe from the results in Table 3, our proposed hybrid CNN-LSTM-based approach outperforms other approaches significantly by producing much lower errors across all metrics. Compared with traditional machine learning methods, deep learning models such as CNN and LSTM have relatively higher prediction accuracy. Hence, we can conclude that CNN-LSTM-based

Table 4. Comparison Results with Models Proposed by Other Researchers

Method	RMSE	Year
SVM [23]	29.82	2013
CNN [2]	18.45	2016
LSTM [7]	17.84	2017
Deep LSTM [12]	16.74	2018
Stacking Ensemble [39]	16.74	2019
The proposed CNN-LSTM	12.76	–

regression approach is better than the standard shallow architecture based regression methods for RUL estimation.

The C-MAPSS FD001 dataset has been extensively studied, and many researchers adopt the dataset to verify their models. Table 4 shows a comparison of the results obtained for the proposed hybrid CNN-LSTM model with the results of other studies in relation to the solution of the problem under consideration. The comparisons demonstrate that the proposed hybrid CNN-LSTM model achieves the best performance and, hence, could have promising application prospects in the prognosis field.

RUL predicted by the proposed hybrid CNN-LSTM model for 100 engines in the testing set is illustrated in Figure 10. This figure indicates that the predicted failure time from our proposed hybrid CNN-LSTM model is very near to the actual failure time or their ground truth values. Along with the running cycles of engines increase (RUL decreases), the prediction error becomes very small, which can be seen in the range of RUL between 0 and 50. This confirms that in the first lifetime stage when the running cycles of engines are small, engines are healthy and have not begun to degrade rapidly. That is the reason prediction error is relatively large in this stage.

High and reliable RUL prediction accuracy at the very end of system lifetime has of course great industrial significance, as this period is critical for PHM applications. However, this evaluation approach could hide the true overall prognostics accuracy, as the prognostics horizon of the algorithm is not considered. The prognostics horizon is critical to achieve trustworthy confidence intervals for the corresponding RUL prediction. These confidence intervals are important due to both inherent uncertainties with the degradation process and potential flaws in the prognosis algorithm [38]. That is why, in this context, the classifier plays an important role—as an identifier of the stage of equipment degradation and it considers prognostics horizon, hence, can be used to ensure the reliability of RUL forecasting.

3.5.2 Network Classifier for Technical Condition Identification. The above prediction results showed that the hybrid CNN-LSTM regressive model gives more accurate forecasts when the value of the RUL is less than 50 cycles. Therefore, to label the data for classification, the time horizon w is set to 50 cycles, then the data will be labeled by two classes. The first class, noted class 0, represents the case where the engine RUL will be greater than the time time horizon w , i.e., $RUL > 50$ (healthy condition). The second class, noted class 1, concerns the case where the engine RUL will not exceed the time horizon w , i.e., $RUL \leq 50$ (faulty condition).

After applying the time-window processing technique, a training set of 17,631 samples and testing set of 10,096 samples are obtained. The resulting training set contains 12,531 samples with “healthy condition” and 5,100 samples with “faulty condition.” The resulting testing set contains 9,210 samples with “healthy condition” and 886 samples with “faulty condition.”

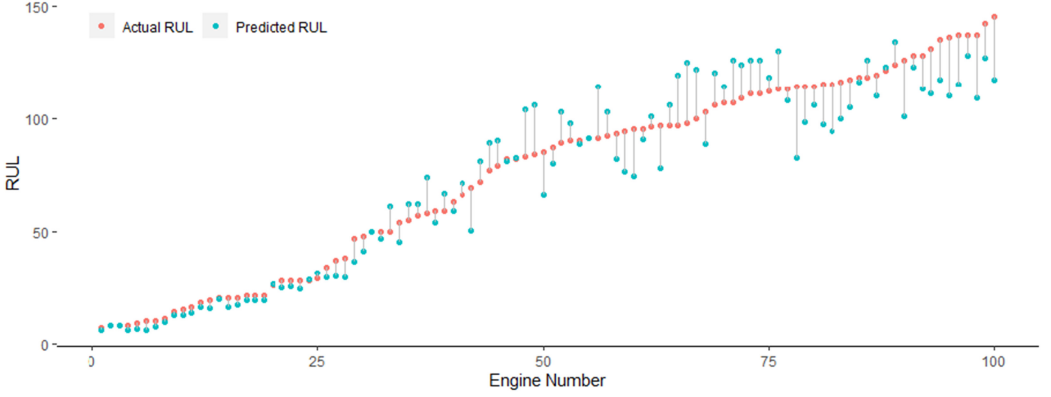


Fig. 10. RUL prediction results of test engines (in ascending order of RUL).

Table 5. Metrics Obtained after Training the Classifier Networks

Class	Precision (%)	Recall (%)	F1-score (%)
class 0	98.33	99.55	98.94
class 1	95.37	84.42	89.56

Once the hybrid CNN-LSTM classifier has been trained on training set, the model is applied to the testing set, obtaining the overall accuracy of 98.05 %. Table 5 shows the classification accuracy for each state class. It can be seen in Table 5 that this network classifier renders better results when classifying engines with more than 30 remaining life cycles (partly because this class has a larger number of instances).

3.5.3 Final Prognostics Schema. The final stage in the workflow of the proposed framework consists in predicting the class to which an engine belongs. For this purpose, a series of measurements from different sensors are introduced into the network classifier. Subsequently, if the above classification is class 1, then the number of remaining cycles is predicted. Algorithm 2 illustrates the process of evaluating performance of the final prognostics schema.

ALGORITHM 2: Final Model Performance Evaluation

Functions: Error metrics MAE, RMSE, and MAPE.

Predicted_class = CNN-LSTM_Classifier($X_{t_{i-N_L+1}:t_i}$)

If predicted_class != 0 **then**

Predicted_rul = CNN-LSTM_Regressor($X_{t_{i-N_L+1}:t_i}$)

errors = MAE(*), RMSE(*), and MAPE(*), where * denotes $|predicted_rul - true_rul|$

End if

Return errors

End function

The application of this workflow to the test set and the errors MAE, RMSE, and MAPE are then, respectively, 3.04, 4.09, and 13.22. The obtained prediction results and the proposed hybrid CNN-LSTM models can be used in intelligent predictive maintenance for the purpose of highly reliable identification of the degradation stage and RUL estimation of complex multi-object systems.

4 CONCLUSIONS

In this article, a new effective data-driven prognostic framework based on hybrid deep learning CNN-LSTM network models was presented. The proposed hybrid network models are divided into two major parts, which are convolutional and recurrent neural network. CNN and LSTM are applied to local patterns and capture temporal long-term dependencies, respectively. In addition, the fully connected network is used to decode the features and information from the CNN-LSTM. When predicting the RUL of a multi-object system the most common procedure is to perform a regression with the set of historical measurements of the different sensors of the system. The main problem with this type of procedure is that if the range of the number of life cycles is too wide, then large errors can occur when performing the regression. This is why the proposed framework consists of two trained hybrid CNN-LSTM networks. The first one is a classifying network that predicts the class of a system according to its state. The second network is regressive and allows to accurately calculate the number of RUL. It provides a better forecast result than using just the single regressive network model.

The proposed hybrid CNN-LSTM models can automatically extract significant features directly from “raw” multi-sensor time series with simple pre-processing for failure prognostics of complex multi-object systems in CPS environment. The experiments are performed on the publicly available C-MAPSS dataset. In the experimental study, the proposed hybrid CNN-LSTM regressive network model is compared to different existing approaches as well as recent studies in the literature in RUL prediction task, the results obtained by the proposed method achieve best overall performance. The modification herein proposed to the hybrid CNN-LSTM regressive architecture allowed the classification with high accuracy. The technical conditions of the equipment are identified with the probability of 99%. The obtained results and algorithms can be used in predictive maintenance systems aimed at reliable identification of the equipment degradation current stage and RUL estimation.

In future work, we plan to apply the proposed data-driven prognostic based on hybrid deep learning models in predictive maintenance tasks and continue to optimize our models to get better performance.

ACKNOWLEDGMENTS

Authors thank reviewers for useful remarks that improve the article.

REFERENCES

- [1] Osama Abdeljaber, Onur Avci, Serkan Kiranyaz, Moncef Gabbouj, and Daniel J. Inman. 2017. Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks. *J. Sound Vibr.* 388 (2017), 154–170.
- [2] Giduthuri Sateesh Babu, Peilin Zhao, and Xiao-Li Li. 2016. Deep convolutional neural network based regression approach for estimation of remaining useful life. In *International Conference on Database Systems for Advanced Applications*. Springer, Cham, 214–228. DOI : https://doi.org/10.1007/978-3-319-32025-0_14
- [3] Yoshua Bengio and Olivier Delalleau. 2011. On the expressive power of deep architectures. In *International Conference on Algorithmic Learning Theory*. Springer, Berlin, 18–36. DOI : https://doi.org/10.1007/978-3-642-24412-4_3
- [4] Guido Dartmann, Houbing Song, and Anke Schmeink. 2019. *Big Data Analytics for Cyber-physical Systems: Machine Learning for the Internet of Things*. Elsevier.
- [5] Yann Dauphin, Harm Vries, Junyoung Chung, and Yoshua Bengio. 2015. RMSProp and equilibrated adaptive learning rates for nonconvex optimization. *arXiv preprint arXiv:1502.04390* (2015), 10.
- [6] Jason Deutsch and David He. 2017. Using deep learning-based approach to predict remaining useful life of rotating components. *IEEE Trans. Syst., Man, Cyber.: Syst.* 48, 1 (2017), 11–20.
- [7] Dong Dong, Xiao-Yang Li, and Fu-Qiang Sun. 2017. Life prediction of jet engines based on LSTM-recurrent neural networks. In *Prognostics and System Health Management Conference (PHM-Harbin)*. IEEE, 1–6. DOI : <https://doi.org/10.1109/PHM.2017.8079264>

- [8] Olga Fink. 2020. Data-driven intelligent predictive maintenance of industrial assets. In *Women in Industrial and Systems Engineering*. Springer, 589–605.
- [9] Yiming Guo, Yifan Zhou, and Zhisheng Zhang. 2021. Fault diagnosis of multi-channel data by the CNN with the multilinear principal component analysis. *Measurement* 171 (2021), 108513.
- [10] Sepp Hochreiter. 1991. *Untersuchungen zu Dynamischen Neuronalen Netzen [in German]*. Diploma thesis. TU Munich.
- [11] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9, 8 (1997), 1735–1780.
- [12] Che-Sheng Hsu and Jehn-Ruey Jiang. 2018. Remaining useful life estimation using long short-term memory deep learning. In *IEEE International Conference on Applied System Invention (ICASI)*. IEEE, 58–61. DOI: <https://doi.org/10.1109/ICASI.2018.8394326>
- [13] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [14] Mathias Kraus and Stefan Feuerriegel. 2019. Forecasting remaining useful life: Interpretable deep learning approach via variational Bayesian inferences. *Decis. Supp. Syst.* 125 (2019), 1–13. DOI: <https://doi.org/10.1016/j.dss.2019.113100>
- [15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444. DOI: <https://doi.org/10.1038/nature14539>
- [16] Jay Lee, Hossein Davari Ardakani, Shanhu Yang, and Behrad Bagheri. 2015. Industrial big data analytics and cyber-physical systems for future maintenance & service innovation. *Procedia CIRP* 38 (2015), 3–7.
- [17] Jay Lee, Moslem Azamfar, Jaskaran Singh, and Shahin Siahpour. 2020. Integration of digital twin and deep learning in cyber-physical systems: Towards smart manufacturing. *IET Collab. Intell. Manuf.* 2, 1 (2020), 34–36.
- [18] Jay Lee, Fangji Wu, Wenyu Zhao, Masoud Ghaffari, Linxia Liao, and David Siegel. 2014. Prognostics and health management design for rotary machinery systems—Reviews, methodology and applications. *Mechan. Syst. Sig. Process.* 42, 1–2 (2014), 314–334. DOI: <https://doi.org/10.1016/j.ymssp.2013.06.004>
- [19] Yaguo Lei, Dong Han, Jing Lin, and Zhengjia He. 2013. Planetary gearbox fault diagnosis using an adaptive stochastic resonance method. *Mechan. Syst. Sig. Process.* 38, 1 (2013), 113–124. DOI: <https://doi.org/10.1016/j.ymssp.2012.06.021>
- [20] Xiang Li, Qian Ding, and Jian-Qiao Sun. 2018. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliab. Eng. Syst. Saf.* 172 (2018), 1–11. DOI: <https://doi.org/10.1016/j.res.2017.11.021>
- [21] Alexis Linard and Marcos L. P. Bueno. 2016. Towards adaptive scheduling of maintenance for cyber-physical systems. In *International Symposium on Leveraging Applications of Formal Methods*. Springer, 134–150.
- [22] J. Liu, W. Wang, F. Ma, Y. B. Yang, and C. S. Yang. 2012. A data-model-fusion prognostic framework for dynamic system state forecasting. *Eng. Applic. Artif. Intell.* 25, 4 (2012), 814–823. DOI: <https://doi.org/10.1016/j.engappai.2012.02.015>
- [23] C. Louen, S. Ding, and C. Kandler. 2013. A new framework for remaining useful life estimation using support vector machine classifier. In *Conference on Control and Fault-tolerant Systems (SysTol)*. IEEE, 228–233. DOI: <https://doi.org/10.1109/SysTol.2013.6693833>
- [24] Ahmed Mosallam, Kamal Medjaher, and Noureddine Zerhouni. 2013. Nonparametric time series modelling for industrial prognostics and health management. *Int. J. Adv. Manuf. Technol.* 69 (2013), 1685–1699. DOI: <https://doi.org/10.1007/s00170-013-5065-z>
- [25] R. Müller, S. Narciss, and L. Urbas. 2017. Interfacing cyber-physical production systems with human decision makers. In *Cyber-physical systems: Foundations, Principles and Applications*. Elsevier, 145–160.
- [26] Kim-Anh Nguyen, Phuc Do, and Antoine Grall. 2015. Multi-level predictive maintenance for multi-component systems. *Reliab. Eng. Syst. Saf.* 144 (2015), 83–94. DOI: <https://doi.org/10.1016/j.res.2015.07.017>
- [27] Khanh T. P. Nguyen and Kamal Medjaher. 2019. A new dynamic predictive maintenance framework using deep learning for failure prognostics. *Reliab. Eng. Syst. Saf.* 188 (2019), 251–262. DOI: <https://doi.org/10.1016/j.res.2019.03.018>
- [28] P. J. García Nieto, E. García-Gonzalo, F. Sánchez Lasheras, and F. J. de Cos Juez. 2015. Hybrid PSO–SVM-based method for forecasting of the remaining useful life for aircraft engines and evaluation of its reliability. *Reliab. Eng. Syst. Saf.* 138 (2015), 219–231. DOI: <https://doi.org/10.1016/j.res.2015.02.001>
- [29] Tom O'Malley. 2020. Hyperparameter tuning with Keras Tuner. Retrieved from <https://blog.tensorflow.org/2020/01/hyperparameter-tuning-with-keras-tuner.html>.
- [30] Sangram Patil, Aum Patil, Vishwadeep Handikherkar, Sumit Desai, Vikas M. Phalle, and Faruk S. Kazi. 2018. Remaining useful life (RUL) prediction of rolling element bearing using random forest and gradient boosting technique. In *ASME International Mechanical Engineering Congress and Exposition*. ASME, 1–7. DOI: <https://doi.org/10.1115/IMECE2018-87623>
- [31] Ron J. Patton, Paul M. Frank, and Robert N. Clark (Eds.). 2013. *Issues of Fault Diagnosis for Dynamic Systems*. Springer Science & Business Media.
- [32] Michael Pecht and Rubycja Jaai. 2010. A prognostics and health management roadmap for information and electronics-rich systems. *Microelectron. Reliab.* 3, 4 (2010), 317–323. DOI: <https://doi.org/10.1016/j.microrel.2010.01.006>
- [33] Huihui Qiao, Taiyong Wang, Peng Wang, Shibin Qiao, and Lan Zhang. 2018. A time-distributed spatiotemporal feature learning method for machine health monitoring with multi-sensor time series. *Sensors* 18, 9 (2018), 1–20. DOI: <https://doi.org/10.3390/s18092932>

- [34] Ajit Kumar Rout, P. K. Dash, Rajashree Dash, and Ranjeeta Bisoi. 2017. Forecasting financial time series using a low complexity recurrent neural network and evolutionary learning approach. *J. King Saud Univ.-Comput. Inf. Sci.* 29, 4 (2017), 536–552.
- [35] Abhinav Saxena, Kai Goebel, Don Simon, and Neil Eklund. 2008. Damage propagation modeling for aircraft engine run-to-failure simulation. In *International Conference on Prognostics and Health Management*. IEEE, 1–9. DOI: <https://doi.org/10.1109/PHM.2008.4711414>
- [36] Maxim Shcherbakov, Artem Glotov, and Sergey Cheremisinov. 2020. Proactive and predictive maintenance of cyber-physical systems. In *Cyber-Physical Systems: Advances in Design & Modelling*, Alla Kravets, Alexander Bolshakov, and Maxim Shcherbakov (Eds.). Springer, Cham, 263–278. DOI: https://doi.org/10.1007/978-3-030-32579-4_21
- [37] Maxim Vladimirovich Shcherbakov, Adriaan Brebels, Nataliya Lvovna Shcherbakova, Anton Pavlovich Tyukov, Timur Alexandrovich Janovsky, Valeriy Anatol'evich Kamaev, et al. 2013. A survey of forecast error measures. *World Appl. Sci. J.* 24, 24 (2013), 171–176.
- [38] J. Z. Sikorska, M. Hodkiewicz, and L. Ma. 2011. Prognostic modelling options for remaining useful life estimation by industry. *Mechan. Syst. Sig. Process.* 25, 5 (2011), 1803–1836. DOI: <https://doi.org/10.1016/j.ymssp.2010.11.018>
- [39] Sandip Kumar Singh, Sandeep Kumar, and J. P. Dwivedi. 2019. A novel soft computing method for engine RUL prediction. *Multim. Tools Applic.* 78, 4 (2019), 4065–4087. DOI: <https://doi.org/10.1007/s11042-017-5204-x>
- [40] Houbing Song, Danda B. Rawat, Sabina Jeschke, and Christian Brecher. 2016. *Cyber-physical Systems: Foundations, Principles and Applications*. Morgan Kaufmann.
- [41] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15 (2014), 1929–1958.
- [42] Sai Van Cuong and Prof Maxim Shcherbakov. 2019. PdM: A predictive maintenance modeling tool implemented as R-package and web-application. In *10th International Symposium on Information and Communication Technology*. ACM, New York, NY, 433–440. DOI: <https://doi.org/10.1145/3368926.3369693>
- [43] Paul J. Werbos. 1988. Generalization of backpropagation with application to a recurrent gas market model. *Neural Netw.* 1, 4 (1988), 339–356.
- [44] Maciej Wielgosz, Andrzej Skoczeń, and Matej Mertik. 2017. Using LSTM recurrent neural networks for monitoring the LHC superconducting magnets. *Nucl. Instrum. Meth. Phys. Res. Sect. A: Accel., Spectrom., Detect. Assoc. Equip.* 867 (2017), 40–50.
- [45] Qianhui Wu, Keqin Ding, and Biqing Huang. 2020. Approach for fault prognosis using recurrent neural network. *Journal of Intelligent Manufacturing* 31, 7 (2020), 1621–1633.
- [46] Ahmed Elsheikh, Soumaya Yacout, and Mohamed-Salah Ouali. 2019. Bidirectional handshaking LSTM for remaining useful life prediction. *Neurocomputing* 323 (2019), 148–156.
- [47] Yuting Wu, Mei Yuan, Shaopeng Dong, Li Lin, and Yingqi Liu. 2018. Remaining useful life estimation of engineered systems using vanilla LSTM neural networks. *Neurocomputing* 275 (2018), 167–179.
- [48] Rui Yang, Mengjie Huang, Qidong Lu, and Maiying Zhong. 2018. Rotating machinery fault diagnosis using long-short-term memory recurrent neural network. *IFAC-Papers OnLine* 51, 24 (2018), 228–232.
- [49] Mei Yuan, Yuting Wu, and Li Lin. 2016. Fault diagnosis and remaining useful life estimation of aero engine using LSTM neural network. In *IEEE International Conference on Aircraft Utility Systems (AUS)*. IEEE, 135–140. DOI: <https://doi.org/10.1109/AUS.2016.7748035>
- [50] Ansi Zhang, Honglei Wang, Shaobo Li, Yuxin Cui, Zhonghao Liu, Guanci Yang, and Jianjun Hu. 2018. Transfer learning with deep recurrent neural networks for remaining useful life estimation. *Appl. Sci.* 8, 12 (2018), 2416.
- [51] Jianjing Zhang, Peng Wang, Ruqiang Yan, and Robert X. Gao. 2018. Deep learning for improved system remaining life prediction. *Procedia CIRP* 72 (2018), 1033–1038.
- [52] Jianjing Zhang, Peng Wang, Ruqiang Yan, and Robert X. Gao. 2018. Long short-term memory for machine remaining life prediction. *J. Manuf. Syst.* 48 (2018), 78–86.
- [53] Guangquan Zhao, Guohui Zhang, Qiangqiang Ge, and Xiaoyong Liu. 2016. Research advances in fault diagnosis and prognostic based on deep learning. In *Prognostics and System Health Management Conference (PHM-Chengdu)*. IEEE, 1–6. DOI: <https://doi.org/10.1109/PHM.2016.7819786>
- [54] Haitao Zhao, Shaoyuan Sun, and Bo Jin. 2018. Sequential fault diagnosis based on LSTM neural network. *IEEE Access* 6 (2018), 12929–12939.
- [55] Shuai Zheng, Kosta Ristovski, Ahmed Farahat, and Chetan Gupta. 2017. Long short-term memory network for remaining useful life estimation. In *IEEE International Conference on Prognostics and Health Management (ICPHM)*. IEEE, 88–95. DOI: <https://doi.org/10.1109/ICPHM.2017.7998311>
- [56] Patrick Zschech, Kai Heinrich, Raphael Bink, and Janis S. Neufeld. 2019. Prognostic model development with missing labels. *Bus. Inf. Syst. Eng.* 61, 3 (2019), 327–343. DOI: <https://doi.org/10.1007/s12599-019-00596-1>

Received August 2020; revised July 2021; accepted September 2021