J Fail. Anal. and Preven. (2021) 21:1596–1604
https://doi.org/10.1007/s11668-021-01205-8

TECHNICAL ARTICLE—PEER-REVIEWED

A New Hybrid Model for RUL Prediction through Machine Learning

Zahra Esfahani · Karim Salahshoor · Behnam Farsi · Ursula Eicker

Submitted: 18 April 2021 / in revised form: 15 June 2021 / Accepted: 12 July 2021 / Published online: 24 July 2021

Abstract Remaining useful life (RUL) prediction plays a significant role in prognostics and health management systems. While three different approaches have been utilized to estimate the RUL, hybrid-based methodologies yield more accurate results in this field. This study aims to introduce a hybrid prognostic approach based on deep learning methods, including long short-term memory (LSTM) and convolutional neural network (CNN). In most of the combined models, CNN is using to extract the features, and then, these LSTM be fed by extracted information from CNN, but in the hybrid model, both LSTM and CNN use organically to enhance the prediction ability. Besides, the time window (TW) is utilized to provide sequential data by sliding it on input data. To evaluate the proposed model's accuracy and speed, the KPCA algorithm is used to determine the dependency of the model on extracted features. The proposed model is validated on the data developed by NASA's commercial modular aero-propulsion system simulation (C-MAPSS) dataset. The results have illustrated that removing less important features has no effect on the proposed model.

Z. Esfahani (✉)
Electrical and Computer Engineering, Islamic Azad University South Tehran Branch, Tehran, Iran
e-mail: zahra.esfahani@ieee.org

K. Salahshoor
Control and Instrumentation, Petroleum University of Technology, Tehran, Iran

B. Farsi
Concordia University, Montréal, Canada

U. Eicker
Concordia University, Civil and Environmental Engineering, Montreal, Canada

Keywords Remaining useful life · Hybrid-based approach · LSTM-CNN · Turbofan engine

Introduction

Condition-based maintenance is an intelligent prognostics approach that has a strong ability to increase reliability and performance in industries. Condition monitoring plays a major role in prognostics and health management (PHM) strategies, and more accurate estimations can be achieved based on this type of maintenance. Model-based, data-based, and hybrid-based approaches are the three main methodologies in prognostics in which fusion of these approaches attracts more research in the latest articles. Due to the capability of data-driven approaches, they have curious attention in RUL prediction. One of the main challenges among data-driven approaches is developing an applicable method to figure out the best relationship between the remaining useful life (RUL) and monitoring data of a system or component. On the other hand, how to read sensor measurements is effective on estimation, and the more available historical data are more momentous.

Long short-term memory (LSTM) networks [1], as a specific division of recurrent neural networks (RNN), have the ability to learn long-term beliefs. These networks remember information for long durations by flowing information within their cells and do not have the time dependency problem based on this default attitude. This feature may influence future estimations. One of the challenging problems of traditional LSTM techniques is that they only utilize the onward direction of input data. Hence, bidirectional LSTM (BiLSTM) has a cell for sequence learning and would take full information of sensor data in both backward and forward directions. Authors in [2] suggested an LSTM approach for distinct faults, and the comparison of the proposed method with a traditional recurrent neural network indicates a high performance of the LSTM network. Furthermore, by using more LSTM layers, feed-forward neural network layers, and output layers, a better realization than traditional networks is achieved [3].

Moreover, by performing a variation in the LSTM network and having a new network called vanilla LSTM, more accurate estimation is provided in different operational conditions and noisy environments [4]. The transfer learning algorithms are widely approved to learn from transferring source data to the target one. A transfer learning technique with a deep neural network is developed for RUL estimation and has an acceptable prediction result with a limited number of instances [5]. The authors considered different distributions for various operating conditions and fault modes and proposed a prognostics method based on LSTM to predict a more reliable and accurate lifetime [6]. A novel methodology for uncertainty modeling using the Wiener process is proposed in [7] in which the LSTM network plays as an estimator and indicates acceptable results in near-failure prediction.

Nevertheless, there were some limitations in utilizing one of LSTM or CNN individually. Therefore, a combination of those approaches as a hybrid methodology called CNN-LSTM has been proposed in the latest research [8–10]. These hybrid approaches' main goal is to improve the estimation performance by integrating the profits of various methods through their combination. An integration of LSTM and CNN is developed in [11], which used an explicit health index and pre-selected data conclusively for feature extraction. The health index is obtained based on regression methods to have a single signal. Then, a fully connected neural network is applied to yield the RUL prognostics model. Two deep learning methods are utilized sequentially in [12] that temporal and spatial features can be extracted by LSTM and CNN, respectively. This hybrid method can diminish noise effects, and therefore, the accuracy of prediction is enhanced. However, among these studies, the lack of an interpretable model and analysis is apparent. In this paper, we proposed a hybrid LSTM-CNN model to predict the RUL of the engines. The proposed model tries to predict each RUL of engines in each cycle. This procedure helps to interpret the model better and understand what is precisely happens during each cycle. Moreover, by using sliding a time window (TW) over our data, our prediction problem converted to a short-term time series prediction. Kernel principal component analysis (KPCA) has been applied to find out whether proposed model is able to perform appropriately or not, while some features have been removed from data.

High reliability and safety are two main issues that should be considered in modern aeronautical technology. Moreover, the turbofan engine plays a vital role in this field, and its performance evaluation will help to better degradation trend estimation and prediction of remaining useful life. Based on these challenges, there are some deficiencies in the existing research on using deep learning methods for RUL prediction. Most of the existing techniques used extracted features from the CNN part as LSTM input. In this regard, the paper proposes a novel hybrid approach based on LSTM-CNN, which performs LSTM and CNN in parallel mode for RUL prediction of turbofan engines.

The main contributions of this work come from the following two attitudes. In the first step, the KPCA approach is utilized for dimension reduction. In the second part, CNN and LSTM used input variables individually. Then, those networks' output is merged to enter the fully connected layer to obtain a more accurate prediction before and after dimension reduction. To better understand the proposed approach, a flowchart of the process is indicated in Fig. 1. To the best of our knowledge, our proposed LSTM-CNN technique is the first one that is based on the parallel use of LSTM and CNN to reach this goal.

The proposed method is validated using C-MAPSS data set. Our results are compared with LSTM and CNN approaches individually and indicates acceptable accuracy and acceptable computation time.

The rest of this paper is organized as follows. In Sect. "Background of Study", we propose a background of study which includes the basic of LSTM and CNN networks. In Sect. "Case Study", the simulated turbofan engine dataset is presented as a case study of the work. Evaluation metrics and implementation are used in Sects. "Evaluation

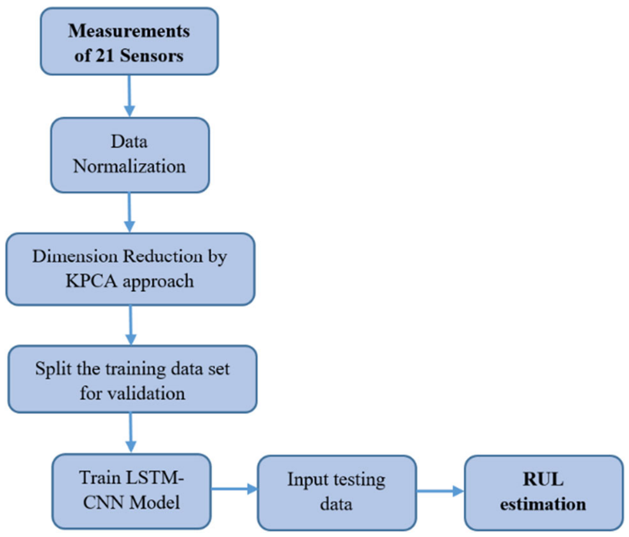
Measurements of 21 Sensors
Data Normalization
Dimension Reduction by KPCA approach
Split the training data set for validation
Train LSTM-CNN Model
Input testing data
RUL estimation

Fig. 1 Flowchart of the proposed process

Metrics" and "Implementation" to indicate the developed technique. Finally, conclusions and future work are involved in Sect. "Conclusion".

Background of Study

Long Short-Term Memory (LSTM)

Recurrent neural networks (RNNs), as a member of the artificial intelligence family, are able to utilize both prior and current data to make estimations. However, the main limitations of these networks are their weakness in long-term prediction and vanishing gradient problems. Therefore, LSTM as a specific architecture of RNN is introduced in [1] for the first time to solve these problems. The main difference between RNN and LSTM topologies is in the inner cell. In general, LSTM is the combination of three cells which are known as the input gate ($i[t]$), forget gate ($f[t]$), and the output gate ($O[t]$). In LSTM, a cell memory is added to store information. The forget gate controls the flow of information among LSTM cells to prevent long-term dependency. The last gate is the output gate which controls the output flow to specify which part of the cell should be utilized for the rest of the network.

The following equations indicate the mathematical process of LSTM:

$$i[t] = \psi(W_i * h[t-1] + b_i]$$

$$f[t] = \psi\left(W_f * h[t-1] + b_f\right]$$

$$O[t] = \psi(W_o * h[t-1] + b_o]$$

$$C[t] = f[t] \odot C[t-1] + i[t] \odot (\phi(W_c * h[t-1] + b_c]$$

$$h[t] = \phi(C[t]) \odot O[t]$$

in which C_t and X_t represent the tth cell and input of the network at time t, respectively. Furthermore, W_i, W_f, and W_o are parameters that should be learned, while b_i, b_f, b_o, and b_c are biased vectors. ϕ is a nonlinear function, ψ is a sigmoid function, and $\odot$ is utilized as a symbol for Hadamard product.

Figure 2 indicates LSTM network architecture which illustrates that each prediction at time t depends on all previous estimations and the learned information from them.

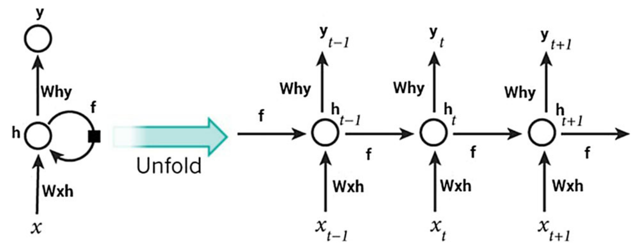

Fig. 2 The basic architecture of the LSTM network

Convolutional Neural Network (CNN)

CNN is a specific architecture of a feed-forward neural network that can extract helpful features from input. This network usually consists of two different types of modules, including convolution operation and polling layers. The convolution operation in CNN produces output by sliding a kernel w, which is called the weight, across the input data. The main advantage of convolution layers is their ability to share similar weights to prevent the overfitting issue. Besides, pooling operation helps to diminish the computation cost and reduce the input dimension by replacing the output of the network with an outline of neighboring output.

Regarding x as input and w as the kernel where both have n dimensions, the i-th factor in this process will be:

$$Y(i) = \sum_{i,j} x(i-j)w(j)$$

for j from 0 to $k-1$. Figure 3 is a representation of the convolution operation utilized by the CNN path.

Case Study

In this section, the proposed methodology is applied to a simulated turbofan engine dataset which is provided by NASA Prognostics Data Repository, and it is known as C-

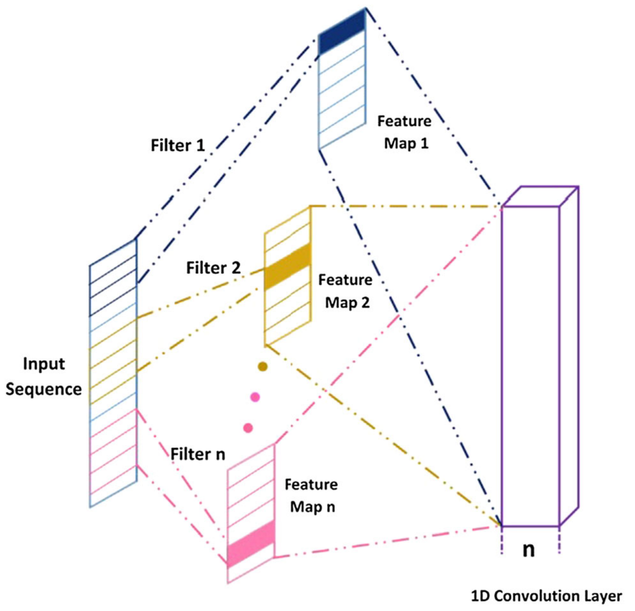

Fig. 3 Convolution operation used by the CNN path [13] *published with permissions

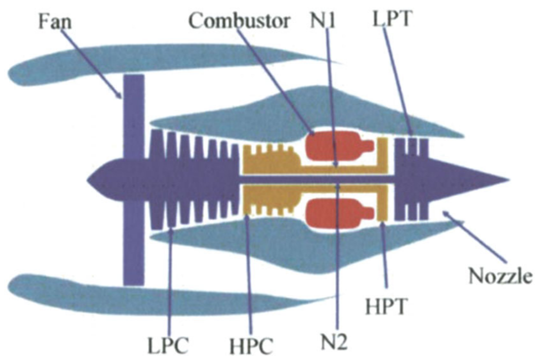

Fig. 4 A simplified diagram of simulated turbofan engine in C-MAPSS [14] *published with permissions

MAPSS Dataset. This simulation is a nonlinear, dynamic, and component-level model that can be helpful for commercial, high-bypass, and dual-spool turbofan engines. The diagram of this simulated turbofan engine is indicated in Fig. 4.

Furthermore, this popular dataset consists of four different datasets, as given in Table 1 that these datasets contain simulated run-to-failure paths of turbofan engines with various operational conditions and fault modes. Furthermore, there are one training dataset and one testing dataset in each sub-dataset. 21 sensors whose details are indicated in Table 2 are utilized to monitor and record run-to-failure data. Each dataset comprises a matrix with an m-by-26 dimension in which m correlates with the number of data points of each turbofan engine. Rows indicate data collected during time cycles, and columns illustrate engine number, operational cycle number, three different operational settings, and 21 sensors values, respectively. These datasets consist of 4 training datasets, four testing datasets, and four files for RUL. The main difference between the training set and testing one is that the last data in the training dataset correspond to the failure time of the engine, but in testing dataset records of sensors has stopped some time before failure and the estimation of remaining useful life of the engine is provided with the testing dataset. On the other hand, the actual values of RUL are provided to certify predicted RUL. In this paper, FD001, which is the first dataset with one fault mode and two operational settings, is utilized to evaluate our proposed hybrid model.

Table 2 Details of 21 sensors for a turbofan engine

Sensor No.	Symbol	Description
1	T2	The total temperature at the fan inlet
2	T24	The total temperature at the LPC outlet
3	T30	The total temperature at the HPC outlet
4	T50	The total temperature at the LPT outlet
5	P2	Pressure at fan inlet
6	P15	The total pressure in bypass-duct
7	P30	Total pressure at HPC outlet
8	Nf	Physical fan speed
9	Nc	Physical core speed
10	epr	Engine pressure ratio (P50/P2)
11	Ps30	Static pressure at HPC outlet
12	phi	The ratio of fuel flow to Ps30
13	NRf	Corrected fan speed
14	NRc	Corrected core speed
15	BPR	Bypass ratio
16	farB	Burner fuel-air ratio
17	htBleed	Bleed enthalpy
18	Nf_dmd	Demanded fan speed
19	PCNfR_dmd	Demanded corrected fan speed
20	w31	HPT coolant bleed
21	w32	LPT coolant bleed

In our work, we analyzed the accuracy and performance of the proposed method using the first dataset (FD001) which has only one operational condition. However, utilizing some useful techniques such as K-means, these operational conditions would be separated to some extent. Because the operation condition separates sensors into different clusters for each data points [15].

Evaluation Metrics

Due to the importance of performance evaluation for assessing predicted models, this section provides different performance metrics to evaluate prognostics algorithms. In

Table 1 Information of available datasets

Datasets		# Fault modes	# Operational conditions	# Train units	# Test units
C-MAPSS	#1	1	1	100	100
	#2	1	6	260	259
	#3	2	1	100	100
	#4	2	6	249	248

general, these metrics can be categorized into accuracy, precision, and robustness. On the other hand, there is a large number of metrics that have been utilized in prognostics. Hence, this paper provides three general metrics especially helpful for the assessment of the error as the difference between the actual output and the target. *R*-squared, root mean-squared error (RMSE), is used in this paper, which can be calculated by the following formula:

$$\mathrm{RMSE} = \sqrt{\left(\frac{1}{N}\right)\sum_{i=1}^{N}(A_i - F_i)^2}$$

where A_i and F_i are the actual and forecasted value of i-th data and N is the number of whole data.

Implementation

Preprocessing

Normalization

The main level of prognostics is the data collection, and after that, data preprocessing level will begin. This process includes two different levels, which are data cleaning and normalization. In FD001 dataset, there is a large difference between the maximum and minimum values of data. Hence, a normalization approach is needed to simplify the analysis. Z-score is a popular normalization technique which is indicated in the following equation:

$$x' = \frac{x - \mu(x)}{\sigma(x)}$$

where x is the raw data of the data set, and $\mu(x)$ and $\sigma(x)$ are the mean value and root mean square deviation of x, respectively. After performing the z-score method, a new dataset with a mean value close to 0 and root mean square deviation close to 1 will be obtained.

Dimensionality Reduction Using KPCA

In general, in the presence of large volumes of data, dimension reduction techniques will be helpful to establish the degradation trend easier. On the other hand, a methodology is needed to capture the nonlinear data pattern. In this article, the kernel principal component analysis (KPCA) is applied. This method is used to reduce PCA limitations in the presence of nonlinear data. The input space x_k is mapped into the feature space F and C^F as the covariate matrix is developed for principal components in the feature space, which is calculated by:

$$C^F = \frac{1}{n}\sum_{i=1}^{n}\phi(x_i)\phi(x_i)^T$$

The nonlinear function is mapped by utilizing a Gaussian radial basis function (RBF) for reducing dimension and splitting different characteristics. This function is represented as follows:

$$G(x_i, x_j) = \exp\left(-\frac{x_i - x_j^2}{2\sigma^2}\right)$$

where σ is a constant parameter, and it is the distribution of Gaussian RBF. Then, the k th feature can be yielded by the following equation:

$$p_k = \sum_{j=1}^{n}\alpha_j^k G(x_j, x)$$

where α_j represents the related quantity of the eigenvector in the covariance matrix.

In our case, after applying KPCA on FD001 data set, according to the variances, sensors [7, 8, 9, 12, 16, 17, 20] and settings [1, 2] were kept, and other features were removed from data set.

Data Preparation

Providing suitable training and test sets as well as targets has remarkable importance for machine learning models. As the main purpose of this paper is that to find RUL for each cycle, time to failure (TTF) for every cycle of each engine is used as the target for the train set, which is calculated through the below equation:

$$\mathrm{TTF}(i) = \max(\mathrm{cycles}) - \mathrm{cycle}(i)$$

However, since TTF for each engine has various lengths, it would be better if, instead of using TTF, fraction TTF (fTTF) be used. fTTF is a kind of scaled TTF that is computed from:

$$\mathrm{fTTF}(i) = \frac{\mathrm{TTF}(i) - \min(\mathrm{TTF})}{\max(\mathrm{TTF}) - \min(\mathrm{TTF})}$$

Creating Image Data

The objective of this section is to transform data into an image time window (TW). By sliding TW over data, these single data will be converted to an image. After removing some sensor which had constant values for all engines, 16 features would be available to be converted into images. In this paper, the shape of images is (50,50). Since we aim to predict fTTF for every cycle, this TW is sliding over every row of train and test set. According to the 16 sensors, the shape of the training set is (15631, 50, 50, 16), and the test set is (8162,50,50,16), whereas after applying KPCA and keeping more important features, the shape of the training set is (15631, 50, 50, 9) and the test set is (8162, 50 ,50 ,9).

Proposed Model

This study presents a new hybrid model, which is a combination of CNN and LSTM. However, some authors study similar models, but the proposed model in this paper has some differences.

In this model, there are two different paths one of which is the LSTM path and the other one is the CNN path. LSTM path contains a flattened layer and LSTM layer in addition to some subtle layers such as Reshape or Repeatvecor. Flatten layer is implemented to convert image data to 1-D data, which is appropriate for the LSTM layer. In the LSTM layer, there are 400 neurons that are activated through ReLU function. After passing some layers like Reshape, the output has a shape of 400.

On the other side, in the CNN path, the approach is a little different. There is no need to use flatten layer at the beginning of the path. Input image data with initial dimension go through from this path. This path begins with 32 neurons in a 2-D convolutional layer, and the kernel size is (2,2). After that, two maxpooling layers with shape (2,2) are implemented to compact the Conv layer's output. In the next layer, another Conv2-D layer is implemented, but this time there are 16 neurons. After passing from the same maxpooling layer, a flatten layer has been used to prepare the output for combination with the output from the LSTM path. The same as the LSTM path, all the neurons in the CNN path have been activated by the ReLU function. Moreover, the CNN path's output is the same shape as the LSTM path's output.

Now, a model is ready to concatenate the outputs from two paths. There is no correlation between the two paths, and data are processed in two different paths, respectively. Concatenate () function is used to merge the outputs. Since there is not any image data, these merged data are processed by a dense layer with 100 neurons to help the next LSTM layer to interpret data more easily. As mentioned, an LSTM layer with 100 neurons has been implemented to interpret the existed sequence in data better. To avoid overfitting, a dropout (0.4) layer and then to prepare data for the final output, a dense layer with five neurons are used. Finally, as this model aims to predict fTTF for each cycle, one dense neuron is implemented to provide the final output. Likewise, other layers in two previous paths, all the layers, have been activated by the ReLU function. Due to the RMSprop [16] characteristics and our model, this optimizer (RMSprop) is used to help the model learn. Also, to reach the best accuracy, some parameters are selected as follows:

batch size = 200

number of epochs = 20

learning rate = 0.001

All the analyses have been implemented in Python 3.8. For the deep learning part, also Keras library with Tensorflow backend was used.

Results

The proposed models are used to predict fTTF of every cycle of each engine; in other words, this model can predict RUL. However, to evaluate this model, two types of training and test data are used; the first model is trained by regular data without any dimensionality reduction; second, it is trained by the output data from KPCA methodology to understand how much this model is stable. In addition, to compare the accuracy, a fully connected LSTM model and fully connected CNN model were implemented. They have been trained by the same data (with and without the KPCA method). Figure 5 shows the loss plot of training the proposed model through data with no KPCA. As it can be seen in the figure, the loss is so close to zero in the last epochs so that in epoch =20, the amount of loss is 0.0035.

Besides, Fig. 6 illustrates predicted and actual fTTF, which seems reasonable. For instance, Fig. 7 shows predicted and actual fTTF of one instance engine with almost 105 cycles. In total, in terms of evaluation of the model, the RMSE is 0.212, and training time took 980 seconds. While the accuracy is reasonable, training time is somehow high.

The proposed model was also evaluated by dimensionality-reduced data. After applying KPCA to data, 16 features were reduced to 9 features. This feature reduction helps the model to train faster. The training time lasted 610 seconds, which is faster than 980s. In addition, the amount of training loss after 20 epochs is 0.0065, which is very

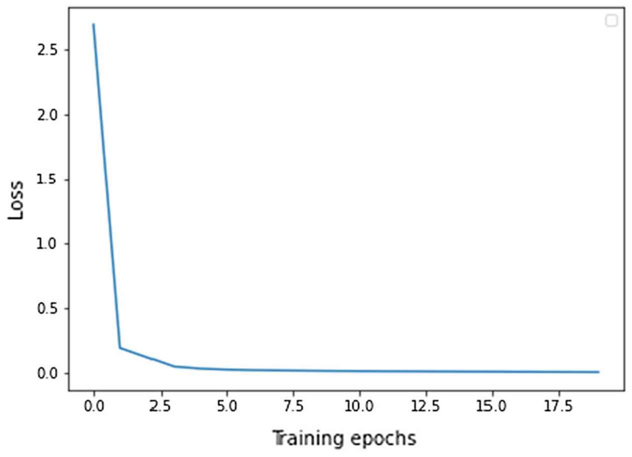

Fig. 5 Training loss plot of the proposed model for regular data

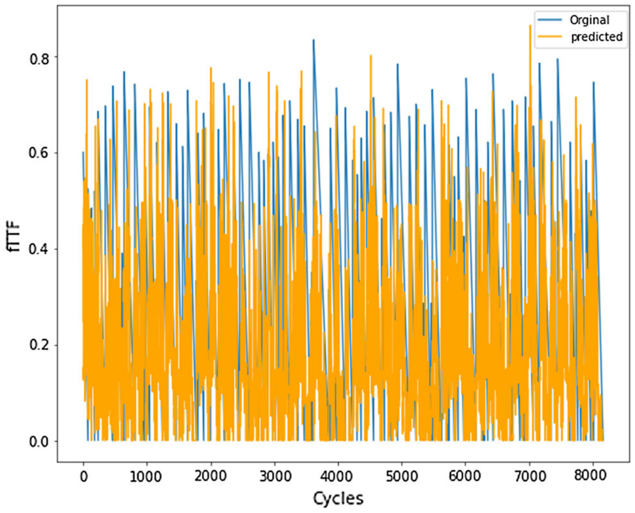

Fig. 6 Predicted and actual data without using KPCA

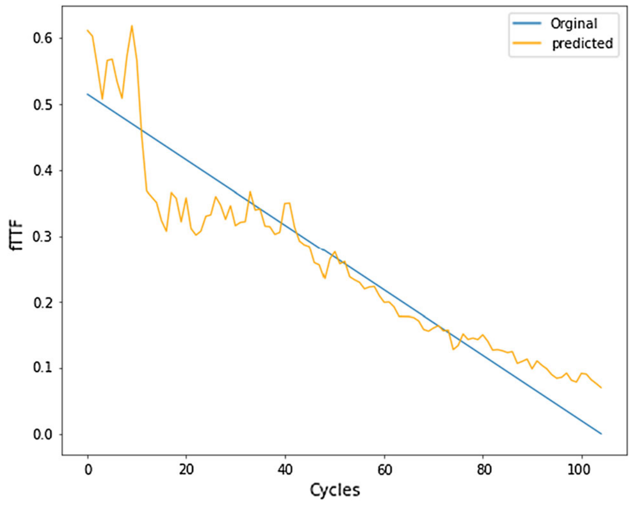

Fig. 7 Predicted and actual data of an engine without using KPCA

Table 3 Different models performance

Dimensionality reduction	Model	RMSE	Training time (S)
KPCA	LSTM	0.455	305
	CNN	0.421	200
	LSTM-CNN	0.225	610
Normal data	LSTM	0.46	505
	CNN	0.44	221
	LSTM-CNN	0.212	980

close to zero, too. Finally, the obtained RMSE is 0.225, which is completely acceptable. All these measurements are provided in Table 3.

Figure 8 indicates training loss of the model, while KPCA data have been applied to it. Figures 9 and 10 also

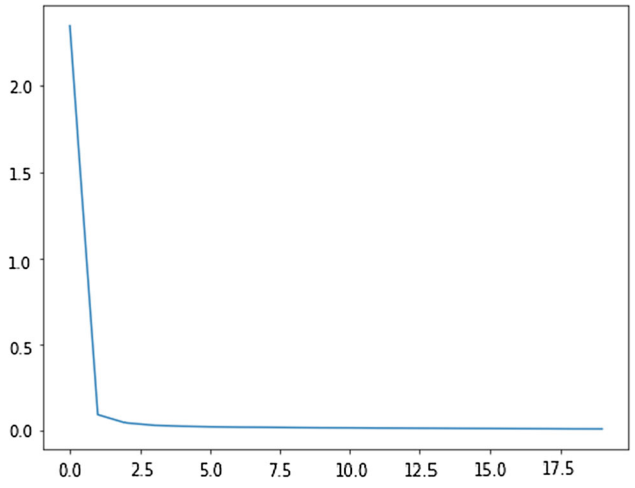

Fig. 8 Training loss plot of the proposed model for reduced data

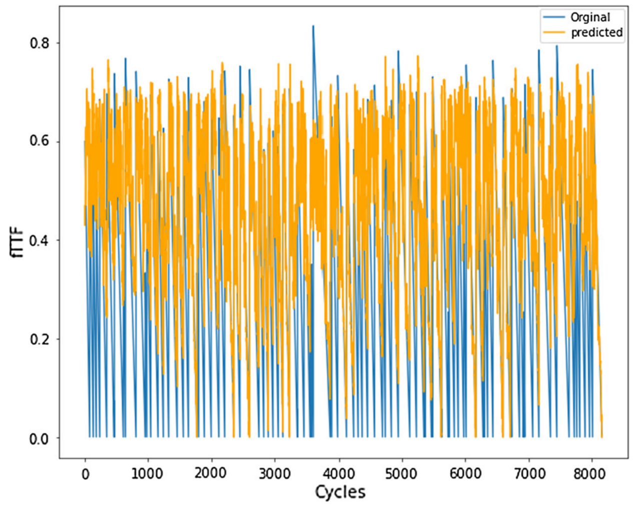

Fig. 9 Predicted and actual data after using KPCA

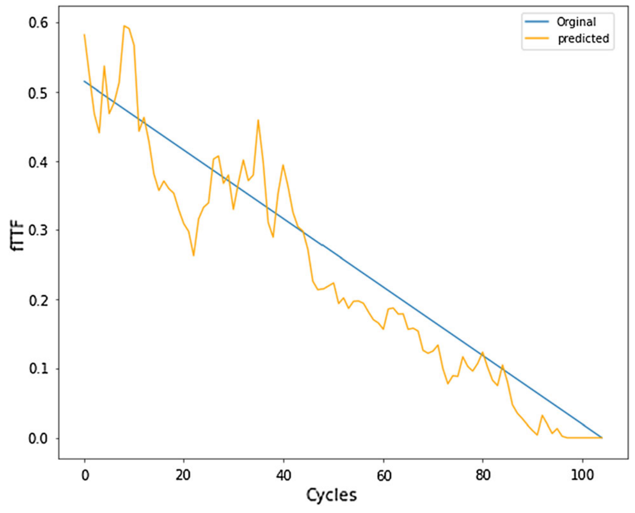

Fig. 10 Predicted and actual data of an engine after using KPCA

illustrate the predicted and actual fTTF of whole engines and one engine, respectively.

Table 3 is a summary of model performance as well as a comparison with other models. As it is shown, the proposed model has more accuracy compared to LSTM and CNN model. KPCA also leads to faster results. It proves that KPCA was able to affect the model positively. Regarding writing models in Table 3, CNN-LSTM indicates the proposed model in that table.

According to the table, while CNN is the fastest model, it is not able to carry out the most accurate prediction. However, regarding the process's final aim, a trade-off between accuracy and speed could be taken place. In total, the proposed model is a powerful model for prediction, and after applying a dimensionality reduction algorithm, it becomes a faster model, while there is not a remarkable negative effect on the accuracy of the model.

Conclusion

In this paper, a novel hybrid prognostics approach based on LSTM-CNN is applied for the RUL prediction of turbofan engines. In order to overcome the problems of high-dimensional data, the KPCA technique is applied at the data level. Due to the sequential nature of our data and also in order to extract the features, CNN and LSTM models were used to improve the accuracy of the model.

The proposed model consists of two different paths for LSTM and CNN which are not correlated. A flattened layer and LSTM layer are utilized to convert data to one-dimensional one. On the other hand, the initial dimension of input data is gone through the CNN path. The outputs of these two paths are used to build the model which can predict the RUL.

It should be put forward that compared with other prognostics approaches, the proposed method has the capability to offer a faster process to obtain an accurate RUL prediction. Moreover, it was approved that this will be helpful to diminish maintenance costs and also better maintenance programming.

Since KPCA proved that it has some good effects on the deep learning model, in future works, we will try our model for more data sets, including FD002, FD003, and FD004. In addition, nowadays, artificial intelligence is becoming more interesting among researchers for being used as a powerful tool for RUL prediction. Therefore, further work will include further discussion on more methodologies of soft computing and using artificial neural networks effectively to enhance the accuracy of our models, while the training time is decreased.

References

1. S. Hochreiter, J. Schmidhuber, Long Short-Term Memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
2. IEEE/CSAA International Conference on Aircraft Utility Systems, Zhongguo hang kong xue hui, Organization of Aviation Utility Systems Engineering, IEEE Control Systems Society, Nanjing Chapter, and Institute of Electrical and Electronics Engineers, *Fault diagnosis and remaining useful life estimation of aero engine using LSTM neural network*. 2016. Accessed: Feb. 01, 2020. [Online]. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=7736343
3. S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, (2017) Long Short-Term Memory Network for Remaining Useful Life estimation, In: *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, Dallas, TX, USA, , pp. 88–95. https://doi.org/10.1109/ICPHM.2017.7998311.
4. Y. Wu, M. Yuan, S. Dong, L. Lin, Y. Liu, Remaining useful life estimation of engineered systems using vanilla LSTM neural networks. Neurocomput. **275**, 167–179 (2018). https://doi.org/10.1016/j.neucom.2017.05.063
5. A. Zhang et al., Transfer learning with deep recurrent neural networks for remaining useful life estimation. Appl. Sci. **8**(12), 2416 (2018). https://doi.org/10.3390/app8122416
6. P. Roberto, de Oliveira da Costa, A. Akçay, Y. Zhang, and U. Kaymak, , Remaining useful lifetime prediction via deep domain adaptation. Reliab. Eng. Syst. Saf. **195**, 106682 (2020). https://doi.org/10.1016/j.ress.2019.106682
7. Y. Deng, A.D. Bucchianico, M. Pechenizkiy, Controlling the accuracy and uncertainty trade-off in RUL prediction with a surrogate Wiener propagation model. Reliab. Eng. Syst. Saf. **196**, 106727 (2020). https://doi.org/10.1016/j.ress.2019.106727
8. T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, (2015) Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks, In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, South Brisbane, Queensland, Australia, Apr. 2015, pp. 4580–4584. https://doi.org/10.1109/ICASSP.2015.7178838.
9. J. Wang, L.-C. Yu, K. R. Lai, and X. Zhang, (2016) Dimensional Sentiment Analysis Using a Regional CNN-LSTM Model. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Berlin, Germany, pp. 225–230. doi: https://doi.org/10.18653/v1/P16-2037.
10. R. Zhao, R. Yan, J. Wang, K. Mao, Learning to monitor machine health with convolutional bi-directional LSTM networks. Sensors. **17**(2), 273 (2017). https://doi.org/10.3390/s17020273
11. Z. Kong, Y. Cui, Z. Xia, H. Lv, Convolution and long short-term memory hybrid deep neural networks for remaining useful life prognostics. Appl. Sci. **9**(19), 4156 (2019). https://doi.org/10.3390/app9194156
12. I. Remadna, S. L. Terrissa, R. Zemouri, S. Ayad, and N. Zerhouni, (2020) Leveraging the Power of the Combination of CNN and Bi-Directional LSTM Networks for Aircraft Engine RUL Estimation. In *2020 Prognostics and Health Management Conference (PHM-Besançon)*, Besancon, France, pp. 116–121. https://doi.org/10.1109/PHM-Besancon49106.2020.00025.

13. A. Al-Dulaimi, S. Zabihi, A. Asif, A. Mohammadi, A multimodal and hybrid deep neural network model for Remaining Useful Life estimation. Comput. Ind. **108**, 186–196 (2019). https://doi.org/10.1016/j.compind.2019.02.004
14. A. Saxena, K. Goebel, D. Simon, and N. Eklund, (2008) Damage propagation modeling for aircraft engine run-to-failure simulation. In: *2008 International Conference on Prognostics and Health Management*, Denver, CO, USA, pp. 1–9. https://doi.org/10.1109/PHM.2008.4711414.
15. S. Pillai, P. Vadakkepat, Two stage deep learning for prognostics using multi-loss encoder and convolutional composite features. Expert Syst. Appl. **171**, 114569 (2021). https://doi.org/10.1016/j.eswa.2021.114569
16. Y. N. Dauphin, H. de Vries, and Y. Bengio, "Equilibrated adaptive learning rates for non-convex optimization," *[cs]*, Aug. 2015, Accessed: Oct. 05, 2020. [Online]. Available: http://arxiv.org/abs/1502.04390arXiv:1502.04390