

A Temporal Pyramid Pooling-Based Convolutional Neural Network for Remaining Useful Life Prediction

Citation for published version (APA):

Song, Y., Bliet, L., Xia, T., & Zhang, Y. (2021). A Temporal Pyramid Pooling-Based Convolutional Neural Network for Remaining Useful Life Prediction. In B. Castanier, M. Cepin, D. Bigaud, & C. Berenguer (Eds.), *Proceedings of the 31st European Safety and Reliability Conference and (ESREL 2021)* (pp. 603-609). Research Publishing (S) Pte Ltd.. https://doi.org/10.3850/978-981-18-2016-8_478-cd

DOI:

[10.3850/978-981-18-2016-8_478-cd](https://doi.org/10.3850/978-981-18-2016-8_478-cd)

Document status and date:

Published: 01/01/2021

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

A Temporal Pyramid Pooling-Based Convolutional Neural Network for Remaining Useful Life Prediction

Ya Song

Department of Industrial Engineering & Innovation Sciences, Eindhoven University of Technology, Netherlands. E-mail: y.song@tue.nl

Laurens Blik

Department of Industrial Engineering & Innovation Sciences, Eindhoven University of Technology, Netherlands. E-mail: l.blik@tue.nl

Tangbin Xia

Department of Industrial Engineering & Management, Shanghai Jiao Tong University, China. E-mail: xtbxtb@sjtu.edu.cn

Yingqian Zhang

Department of Industrial Engineering & Innovation Sciences, Eindhoven University of Technology, Netherlands. E-mail: yqzhang@tue.nl

Abstract: Remaining Useful Life (RUL) prediction is a key issue in Prognostics and Health Management (PHM). Accurate RUL assessments are crucial for predictive maintenance planning. Deep neural networks such as Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) have been widely applied in RUL prediction due to their powerful feature learning capabilities in dealing with high-dimensional sensor data. The sliding time window method with a predefined window size is typically employed to generate data samples to train such deep neural networks. However, the disadvantage of using a fixed-size time window is that we might not be able to apply the resulting predictive model to predict new sensor data whose length is shorter than the predetermined time window size. Besides, as the length of sensor data varies, the traditional unchanged and subjectively set time window size may be inappropriate and impair the prediction model's performance. Therefore, we propose a Temporal Pyramid Pooling-Based Convolutional Neural Network (TPP-CNN) to increase model practicability and prediction accuracy. With the temporal pyramid pooling module, we can generate data samples of arbitrary time window sizes and use them as inputs of CNN. In the training phase, CNN can learn to capture temporal dependencies of different lengths since we feed in samples with different time window sizes. In this novel manner, the learned model can be used to test data with arbitrary sizes, and its predictive ability is also improved. The proposed TPP-CNN model is validated on the C-MPASS turbofan engine dataset, and the experiments have demonstrated its effectiveness.

Keywords: Remaining Useful Life, Deep Learning, Convolutional Neural Network, Temporal Pyramid Pooling, Time Window Size.

1. Introduction

Traditional RUL prediction methods include physics-based approaches and data-driven approaches (Xia et al. 2018). Physics-based approaches build mathematical models to simulate the process of physical degradation, while data-driven approaches perform prediction by capturing the intrinsic trends within condition monitoring data of equipment. With the developments of Internet of Things (IoT) technologies and Artificial Intelligence (AI) algorithms, data-driven approaches have become more convenient and efficient, and they have become more and more popular in the field of PHM (Fink et al. 2020). Compared to traditional data-driven approaches such as statistical models and similarity-based methods, deep learning methods have high nonlinear approximation ability and extensive data processing capability. In industry, they are suitable for processing high-dimensional condition monitoring data. Within deep learning methods, Convolutional Neural Network (CNN) has the characteristics of parameter sharing and sparse interaction, thus it can effectively learn features from sensor data through filters (Li et al. 2018). Long Short-Term Memory (LSTM) is an improved version of the original recurrent neural network, effectively avoiding gradient

disappearance problems by adding the gate mechanism (Zheng et al. 2017). These two types of deep neural networks have been widely applied in RUL prediction, and they have achieved great success (da Costa et al. 2019, Zhao et al. 2020).

When using deep neural networks to predict RUL, we require sufficient run-to-failure sensor data to train a prediction model and then input test data to get the prediction value. These are the two main steps of the supervised learning process. Typically, the sliding time window method with a predefined window size is employed to generate supervised learning data samples. As shown in Figure 1, In the training process, we take sensor data within the previous time window as input features and view the RUL value in the next time step as the output label. As the time window slides, the original sensor data stream is converted into sample-label pairs. Only data within the last time window is fed to the trained model for RUL prediction in the test process. In this data preparation process, the time window size is an essential parameter that needs to be delicately picked. Our previous research showed that the time window size should neither be too large nor too small (Xia et al. 2020). If the time window size is set too large, we might not be able to apply the

trained model to predict RUL for new sensor data whose length is shorter than the predetermined time window size. Thus, the applicability of the trained model is reduced. On the other hand, if the time window size is too small, the trained model can only capture a type of short-term temporal dependencies. And the utilization of the test data is limited, which is also detrimental to RUL prediction. Previous experiments have shown that the time window size critically affects prediction accuracy. In specific problems, the prediction error may be lower when the time window size is larger, and this is because a larger time window will contain more valuable information, facilitating model learning (Li et al. 2018).

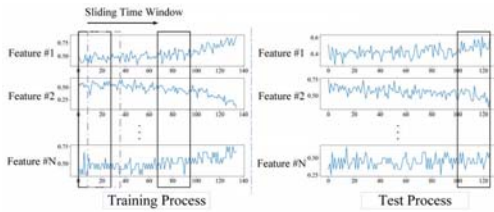


Fig. 1. Sliding time window approach in training and test process

Many advanced deep neural network structures have been applied in RUL predictions. However, few previous researches focus on the time window size optimization. The main challenges are as follows:

- 1) There is no consensus on the optimal time window size setting on the RUL prediction problem, and researchers usually set it subjectively. Though methods proposed in the chaotic time series analysis field like False Nearest Neighbours (FNN) have been applied to reconstruct state space (Ma et al. 2019). These methods are developed based on some hypotheses, and it is hard to employ them on the noise corrupted and high-dimensional condition monitoring data that we are dealing with.
- 2) As we may have multiple run-to-failure data and the length of these time series varies, the optimal time window size of different sensor data may be inconsistent. Even if we obtain the optimal time window size for one short time series sequence, this time window setting may perform poorly on another long sequence. So it is still challenging to determine a single fixed time window size in RUL predictions.
- 3) In predictive maintenance applications, there may be a massive inconsistency among sensor data's length, or these applications require reliable predictions immediately after a particular event occurs. The RUL prediction model needs to be dynamically adapted to different sensor data lengths (Chandra et al. 2018), and the time window size cannot be predetermined.

This paper presents a Temporal Pyramid Pooling-Based Convolutional Neural Network (TPP-CNN) to overcome above problems. This novel method can increase model practicability and prediction accuracy together. With the temporal pyramid pooling module, we can generate data samples of arbitrary time window sizes and

use them as CNN inputs. In the training phase, CNN can learn to capture temporal dependencies of different lengths since we feed in samples with different time window sizes. In the test phase, the trained model can be used to test data with arbitrary sizes, and its predictive ability is also improved. To make full use of the trained model, we obtain multiple test data predictions and ensemble them to improve model robustness. The effectiveness of the TPP-CNN model is validated on the C-MPASS turbofan engine dataset. Experimental results show that it achieves higher accuracy compared with some state-of-the-art methods.

The remainder of this paper is structured as follows: Section 2 presents the proposed TPP-CNN framework. Experimental validation is shown in Section 3. Section 4 illustrates the results comparison and effect analysis. Section 5 provides the conclusions.

2. TPP-CNN

TPP-CNN model mainly consists of convolutional layers and temporal pyramid pooling layers. This section introduces these two types of layers and then describes the training and test procedures.

2.1. Convolutional layer

After the sliding time window processing, CNN's inputs are batches of two-dimensional tensors whose shape is determined by the time window size and the feature dimensions, as shown in Figure 2. In the convolution layer, a set of filters (also called convolution kernels) is used to convolve with the input data to extract local features. The convolution operation is shown in the following equation:

$$f = \varphi(\mathbf{u} * \mathbf{k} + b), \quad (1)$$

where f refers to the feature map generated by the convolution operation, \mathbf{u} and \mathbf{k} represent the input data and the convolution kernel, respectively. φ is the activation function and b is the bias term. Through the convolution operation, the significant features are extracted from the original sensor data.

2.2. Temporal pyramid pooling layer

In 2015, the Spatial Pyramid Pooling (SPP) method was proposed to eliminate the requirement that only fixed-size input images can be fed into a CNN (He et al. 2015). Researchers have analyzed that the fully connected layers that followed CNN need to have a fixed input. By replacing the last traditional pooling layer with the SPP layer, the output representation of CNNs remains constant regardless of the input sample size change so that images with arbitrary sizes can be fed into the networks for training. The one-dimension version of SPP, i.e., Temporal Pyramid Pooling (TPP) was then proposed to tackle the video-based action recognition problems (Wang et al. 2016). In this paper, we apply TPP to remove the fixed time window size constraint of CNNs. Let the input and output dimensions of the TPP layer be (N, C, L_{in}) and (N, C, L_{out}) . Here, N is the batch size. C denotes the number of channels, which is equal to the filter number in

the previous convolutional layer. L_{in} and L_{out} is the length of the input sequence and output sequence, respectively. Firstly, we should set the pyramid level M and the pyramid size n_i in the level i . Then the length of the output sequence is determined by:

$$L_{out} = \sum_{i=1}^M n_i. \quad (2)$$

To achieve the set L_{out} , the parameters of the TPP layer is set as follows:

$$K_{TPP} = S_{TPP} = \left\lfloor \frac{L_{in}}{n_i} \right\rfloor, \quad (3)$$

$$P_{TPP} = \left\lfloor \frac{K_{TPP} \times n_i - L_{in} + 1}{2} \right\rfloor. \quad (4)$$

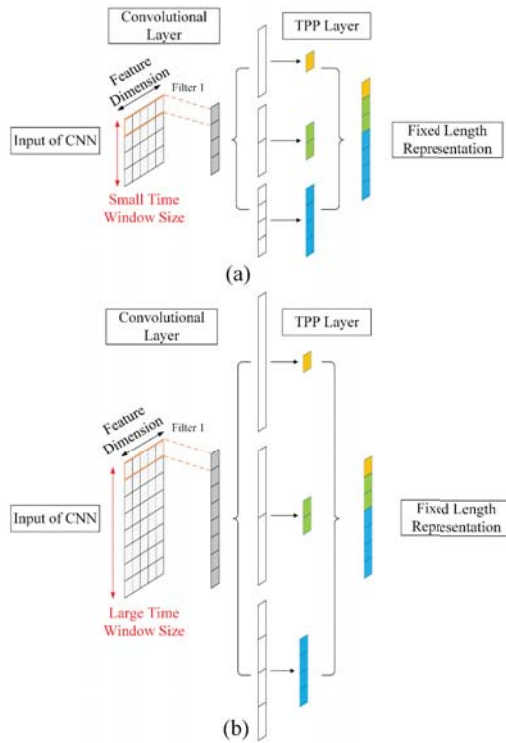


Fig. 2. TPP-CNN framework when (a) the time window size is small and (b) the time window size is large. After processing by a TPP layer with pyramid size $\{1, 2, 4\}$, the length of output representation is uniformly 7.

Here, K_{TPP} , S_{TPP} and P_{TPP} are the kernel size, the stride size, and the padding size of the Maxpooling operation in the TPP layer, respectively. Figure 2 illustrates an example configuration of 3-level pyramid

pooling with the pyramid size $\{1, 2, 4\}$. Thus, the fixed length of output representation is seven. Though the input data size is different, the output representation is always the same size, and this allows us to feed samples with varying time window sizes during the training phase.

2.3. Multiple time windows training phase

In the training phase, we firstly choose several proper time window sizes and process original sensor data with sliding time window approaches. Then the samples are mixed and randomly shuffled. Thus, there is no specific training sequence. At last, we build the TPP-CNN model and begin to train it. By feeding TPP-CNN samples with different time window sizes, it can learn to capture different lengths' temporal dependencies to improve prediction accuracy. To avoid the over-fitting problem, we apply the methods such as dropout and early stopping. The training procedure of TPP-CNN while there are three types of training samples is shown in Figure 3.

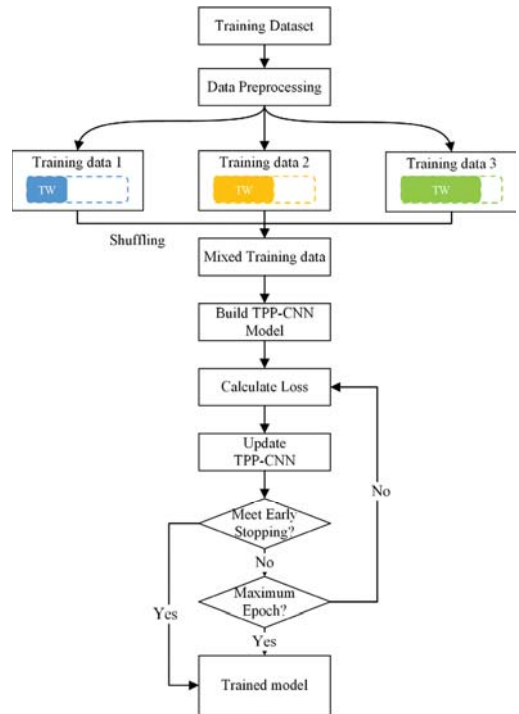


Fig. 3. Multiple time windows training phase

2.4. Multiple time windows test phase

In the test phase, the test units are first sorted from smallest to largest according to their sensor data's length. As the example shown in Figure 4, we then divide test units into multiple subsets like Test data 1, Test data 2, and Test data 3 according to the set time window sizes. The sensor data length of test units in Test data 1 is relatively short. It is unacceptable to apply a larger time window to process sensor data. Only one type of time window is

exploited to generate test samples. While in Test data 3, the lengths of test units are longer, multiple types of time windows can be applied, and thus we can generate several test samples for one test unit. As there are no input size constraints in TPP-CNN, all these test samples can be fed into the trained model to obtain an RUL prediction. Here we apply a simple averaging method to ensemble different prediction values for one test unit to improve model robustness. Finally, the predictions on each test subset are concatenated together to form the final prediction.

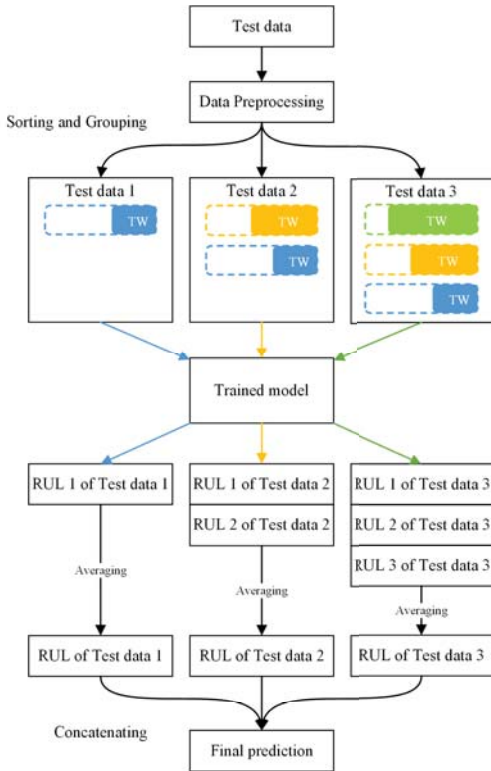


Fig. 4. Multiple time windows test phase

3. Experimental study

In this section, the performance of the proposed TPP-CNN model is empirically evaluated on a public RUL prediction dataset. A description of the dataset is given first, and then details of the experimental procedure are introduced.

3.1. Datasets description

The C-MAPSS turbofan engine dataset is a widely used public dataset in the field of RUL prediction (Saxena et al. 2008). This dataset has four subsets, these subsets are named as FD001, FD002, FD003 and FD004. Each subset is divided into a training set, which records the run-to-failure condition monitoring data of the turbofan engines, and a test set, terminating at a certain point before complete failure. The goal is to predict RUL for test

engines (in terms of the number of running cycles). The condition monitoring data are multivariate time series data and whose total feature dimension is 24. The condition monitoring data is contaminated by noise, and the initial wear state and manufacturing variation of the turbofan engines are unknown, making it difficult to predict the RUL accurately.

The running cycle distribution of the engines in the training and test set of FD001 is shown in Figure 5. We can see that there is a high degree of inconsistency between training and test engines, and the running cycle of training engines is relatively larger. The detailed descriptions of FD001 are presented in Table 1, as the minimum running cycle of training and test engines is 31. In traditional deep learning methods, a fixed time window size should be predefined (Li et al. 2018). This type of fixed time window size should not exceed 31. Otherwise, the trained model is not applicable to some test engines. Previous researchers commonly set the time window size equal to 30 for this dataset (Zheng et al. 2017, Li et al. 2018, da Costa et al. 2020). This small and fixed time window size setting is inappropriate for test engines whose sensor data length is relatively large (Xia et al. 2020).

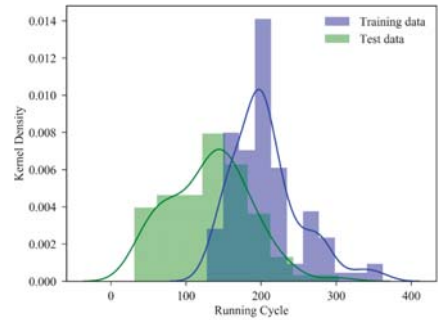


Fig. 5. Running cycle distribution of engines in FD001 (Xia et al. 2020)

Table 1. Description of FD001dataset

FD001	Training set	Test set
Engine number	100	100
Number of data	20631	13096
Minimum running cycle	128	31
Maximum running cycle	362	303

3.2. Experimental procedure

Data pre-processing such as feature selection and feature normalization is performed to apply the proposed model to this data set (Li et al. 2018). Then we use a piece-wise function to rectify the training labels and make them not larger than 125 to avoid overestimating RUL (Xia et al. 2020). Since previous papers chose to rectify test labels either (Ellefsen et al. 2019, da Costa et al. 2020), we also evaluate the proposed model performance when the test labels are rectified. After the data pre-processing, the sliding time window approach is applied to produce samples. In the proposed TPP-CNN model, we generate

four types of samples with different time window sizes, and the time window sizes of these samples are set to be 30, 60, 90, and 120, respectively. After the sliding time window processing, the samples are mixed and fed into TPP-CNN. The detailed network parameters of each layer in the TPP-CNN model are shown in Table 2. The batch size is set to 128, and the maximum training epoch is set to 30. In the TPP layer, we set the pyramid level to be three, and the pyramid sizes in each level are {10, 15, 20}.

Table 2. Parameter setting of TPP-CNN model

Layer Index	Type	Filters/Neurons	Dropout	Activation Function
1	Conv1D	50	0	ReLU
2	Conv1D	50	0	ReLU
3	TPP	{10, 15, 20}	/	/
4	FNN	50	0.5	ReLU
5	FNN	50	0	ReLU
6	FNN	1	0	Linear

The 100 test engines in FD001 are then sorted according to the number of running cycles, and they are divided into four subsets. Figure 6 shows how the test engines in FD001 are grouped according to the number of running cycles. The last subset, Test data 4, contains nearly two-thirds of the test engines. By introducing TPP, the time window can be increased to the appropriate size for these test engines. For the engines in different test subsets, the available time window sizes are selected to generate test samples. For example, the time window size for Test data 1 can only be 30, while in units in Test data 4, the time window size can be 30, 60, 90, and 120. We feed these test samples into trained TPP-CNN and ensemble output values to improve prediction accuracy and robustness.

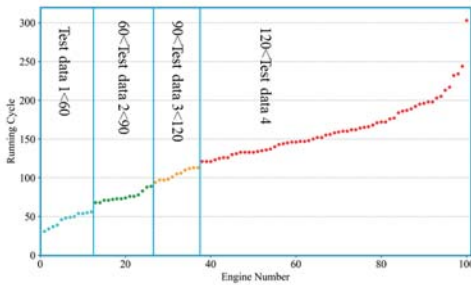


Fig. 6. We divide test engines in FD001 into four subsets according to the number of running cycles. In Test data 1, the engines' running cycles are less than 60, and in Test data 4, the engines' running cycles are larger than 120.

4. Results and effects discussions

4.1. Experimental results comparison

To quantitatively analyze the proposed TPP-CNN model performance, we use Root Mean Squared Error (RMSE) to evaluate RUL prediction accuracy. We run the model 10

times and calculate its mean performance. Table 3 shows the comparison of different models on the first subset FD001, and Table 4 shows RMSE of models on FD002, FD003 and FD004 when we do not rectify test labels. Researchers have widely utilized this dataset to verify their models. Based on the advanced deep neural network structure, researchers often combine other machine learning paradigms such as Transfer Learning (da Costa et al. 2020, Michau et al. 2021) and Semi-supervised Learning (Ellefsen et al. 2019) to enhance model performance. We observe that the proposed TPP-CNN model achieves state-of-the-art performance.

Table 3. Comparison results on FD001 with models proposed by other researchers

Model name	RMSE with rectified test labels	RMSE without rectified test labels
LSTM (Zheng et al. 2017)	16.14	\
MODBNE (Zhang et al. 2016)	\	15.04
ELM (Zheng et al. 2018)	\	13.78
DANN (da Costa et al. 2020)	13.64	\
DCNN (Li et al. 2018)	12.61	13.32
Semi-supervised setup (Ellefsen et al. 2019)	12.56	\
MTW CNN-BLSTM Ensemble (Xia et al. 2020)	\	12.66
Proposed TPP-CNN	11.60	12.64

Table 4. RMSE comparison without rectified test labels on FD002, FD003 and FD004

Model name	FD002	FD003	FD004
MODBNE (Zhang et al. 2016)	25.05	12.51	28.66
Random Forest (Zhang et al. 2016)	29.59	20.27	31.12
CNN (Babu et al. 2016)	30.29	19.82	29.16
DCNN (Li et al. 2018)	24.86	14.02	29.44
Proposed TPP-CNN	25.92	12.39	26.84

4.1. Effect analysis of TPP

To analyze the TPP module's effectiveness, we develop a benchmark model, i.e., the traditional CNN model, to compare the performance with TPP-CNN on FD001. These two models are trained when the test labels are rectified. The parameter setting of this traditional CNN model is shown in Table 5. Except for the non-existence of the TPP layer, the other neural network parameters remain the same as TPP-CNN. As this traditional CNN model's input data can only be fixed, we set the time window size

to be 30, similar to models in existing approaches (Li et al. 2018).

Table 5. Parameter setting of CNN model

Layer Index	Type	Filters/Neurons	Dropout	Activation Function
1	Conv1D	50	0	ReLU
2	Conv1D	50	0	ReLU
3	FNN	50	0.5	ReLU
4	FNN	50	0	ReLU
5	FNN	1	0	Linear

Table 6. Comparison results with CNN

Model name	RMSE Mean	RMSE Std
CNN	12.95	0.30
TPP-CNN	11.60	0.45

Table 6 shows the mean value and the standard deviation (Std) of the RMSE of both two models on FD001. By applying the TPP module, the mean value of RMSE decreases significantly. However, the prediction standard deviation increases, which may be because of new uncertainty introduced by the multiple time windows training and test phase. Then we analyze the performance of two models in each test subset. As shown in Figure 7, in Test data 1, the mean and standard deviation of RMSE raise dramatically after applying the TPP module. This is because the model learns the long-term temporal dependency by enlarging the time window size. This kind of information is invalid for test engines whose sensor data is relatively short, leading to worse prediction performance. In the remaining three test subsets, TPP-CNN can outperform the traditional CNN, and this is due to several reasons as follows. Firstly, the TPP module allows a model to learn long-term temporal dependency, which exists within these test engines with longer tensor data. Secondly, by increasing the time window size, we can take full advantages of the training and test data. At last, by executing ensemble operation in the test phase, the prediction accuracy can be further improved. TPP-CNN has the most significant performance improvement on Test data 4, where there are nearly two-thirds of the test engines, so TPP-CNN can significantly outperform the traditional CNN.

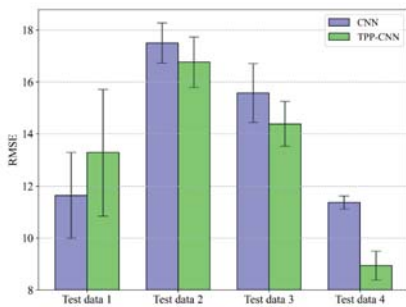


Fig. 7. Comparison in different test subset

5. Conclusions

In this paper, we propose a Temporal Pyramid Pooling-Based Convolutional Neural Network (TPP-CNN) to predict RUL for turbofan engines. By applying temporal pyramid pooling, we remove the fixed time window size constraint of traditional CNNs. Thus, we can generate data samples with different time window sizes and feed them into one TPP-CNN model. In the training phase, TPP-CNN can learn to capture different temporal dependencies within the original sensor data, which improves prediction accuracy. In the test phase, the trained model can be applied to test data with different time window sizes, which improves the model’s practicability and predictive ability. To make full use of the trained model, we obtain multiple predictions for one test time series sequence and ensemble them to improve model robustness. The proposed TPP-CNN model has been validated on the C-MPASS turbofan engine dataset, and the experiments have demonstrated it achieves state-of-the-art results on this public dataset. In the future, we plan to apply surrogate models to optimize the time window size setting and tune the neural network parameters.

References

Xia, T., Dong, Y., Xiao, L., Du, S., Pan, E., & Xi, L. (2018). Recent advances in prognostics and health management for advanced manufacturing paradigms. *Reliability Engineering & System Safety*, 178, 255-268.

Fink, O., Wang, Q., Svensén, M., Dersin, P., Lee, W. J., & Ducoffe, M. (2020). Potential, challenges and future directions for deep learning in prognostics and health management applications. *Engineering Applications of Artificial Intelligence*, 92, 103678.

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

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

da Costa, P. R. D. O., Akeay, A., Zhang, Y., & Kaymak, U. (2019). Attention and long short-term memory network for remaining useful lifetime predictions of turbofan engine degradation. *International Journal of Prognostics and Health Management*, 10, 034.

Zhao, C., Huang, X., Li, Y., & Yousaf Iqbal, M. (2020). A Double-Channel Hybrid Deep Neural Network Based on CNN and BiLSTM for Remaining Useful Life Prediction. *Sensors*, 20(24), 7109.

Xia, T., Song, Y., Zheng, Y., Pan, E., & Xi, L. (2020). An ensemble framework based on convolutional bi-directional LSTM with multiple time windows for remaining useful life estimation. *Computers in Industry*, 115, 103182.

Ma, G., Zhang, Y., Cheng, C., Zhou, B., Hu, P., & Yuan, Y. (2019). Remaining useful life prediction of lithium-ion batteries based on false nearest neighbors and a hybrid neural network. *Applied Energy*, 253, 113626.

Chandra, R., Ong, Y. S., & Goh, C. K. (2018). Co-evolutionary multi-task learning for dynamic time series prediction. *Applied Soft Computing*, 70, 576-589.

- He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9), 1904-1916.
- Wang, P., Cao, Y., Shen, C., Liu, L., & Shen, H. T. (2016). Temporal pyramid pooling-based convolutional neural network for action recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12), 2613-2622.
- Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management* (1-9).
- Zhang, C., Lim, P., Qin, A. K., & Tan, K. C. (2016). Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE transactions on neural networks and learning systems*, 28(10), 2306-2318.
- Zheng, C., Liu, W., Chen, B., Gao, D., Cheng, Y., Yang, Y., ... & Peng, J. (2018). A data-driven approach for remaining useful life prediction of aircraft engines. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 184-189.
- da Costa, P. R. D. O., Akçay, A., Zhang, Y., & Kaymak, U. (2020). Remaining useful lifetime prediction via deep domain adaptation. *Reliability Engineering & System Safety*, 195, 106682.
- Michau, G., & Fink, O. (2021). Unsupervised transfer learning for anomaly detection: Application to complementary operating condition transfer. *Knowledge-Based Systems*, 216, 106816.
- Ellefsen, A. L., Bjørlykhaug, E., Æsøy, V., Ushakov, S., & Zhang, H. (2019). Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering & System Safety*, 183, 240-251.
- Babu, G. S., Zhao, P., & Li, X. L. (2016). Deep convolutional neural network based regression approach for estimation of remaining useful life. *International conference on database systems for advanced applications*, 214-228.