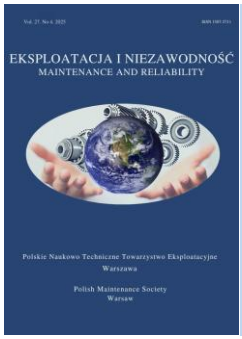


Eksploatacja i Niezawodnosc – Maintenance and Reliability
Volume 27 (2025), Issue 4
journal homepage: http://www.ein.org.pl

Article citation info:
Deng L, Liu J, Liu M, Cao Y, Wen C, Deng X, Remaining Useful Life Prediction Based on Cross-Temporal Dynamic Graph Convolutional Network, Eksploatacja i Niezawodnosc – Maintenance and Reliability 2025: 27(4) http://doi.org/10.17531/ein/203249

Remaining Useful Life Prediction Based on Cross-Temporal Dynamic Graph Convolutional Network

Indexed by:

Liwei Deng[a,b], Jixin Liu[a,b,*], Mei Liu[a], Yue Cao[a], Chenglin Wen[a], Xingchao Deng[b]

[a] School of Automation, Guangdong University of Petrochemical Technology, China
[b] School of Information and Control Engineering, Jilin Institute of Chemical Technology, China

Highlights

- A GNN model for predicting RUL using multi-sensor data is proposed
- A dynamic graph generation method without a priori knowledge is designed
- A decay graph based on spatio-temporal distance is designed

Abstract

Taking advantage of deep learning (DL) to extract hidden degradation signals from machinery monitoring data has led to significant advancements in predicting equipment's remaining useful life (RUL). However, existing methods that use similarity and adaptive adjacency matrices to construct graphs fail to reflect sensor relationships accurately. This article presents a cross-temporal dynamic graph convolutional network (CTDGCN) for RUL prediction to address this issue. The CTDGCN combines cross-temporal modeling with dynamic spatio-temporal graph construction, collecting multi-sensor time series signals to create dynamic graph embeddings. By constructing a cross-temporal sensor network, temporal and spatial features are extracted to design a decay graph based on temporal distance. This model utilizes decay and cross-temporal pooling layers to aggregate information and capture complicated spatio-temporal dependencies. Studies conducted on two cases indicate that the CTDGCN model significantly outperforms existing models in RUL prediction tasks.

Keywords
remaining useful life, cross-temporal, graph convolutional network, time series

1. Introduction

To meet the demands of industrial development, machinery has become increasingly complex, and its performance often deteriorates over time, even leading to failures that can severely obstruct production efficiency and even pose safety risks to workers 1. In terms of prognostics and health management (PHM), the remaining useful life (RUL) is a critical component [2,3]. Traditionally, the maintenance of complex machinery depends on regular manual inspections, however, nowadays, numerous studies aim to achieve precise RUL predictions. Accurate RUL prediction enables the estimation of equipment's normal operational time and facilitates timely maintenance planning, which helps to reduce failures, injuries, and economic costs. Therefore, it is crucial to achieve highly accurate RUL predictions.

Among the studies on RUL, three general approaches can be classified into: physics-based, data-driven, and hybrid-driven

(*) Corresponding author.
E-mail addresses: L. Deng, 1043485394@qq.com, J. Liu (ORCID: 0009-0000-1169-9521) liujixin2000@gdupt.edu.cn, M. Liu, liumei@gdupt.edu.cn, Y. Cao, caoyue2000@gdupt.edu.cn, C. Wen (ORCID: 0009-0000-6391-1168) wencl@gdupt.edu.cn, X. Deng, 1466309745@qq.com,

approaches 4. Physics-based methods offer strong interpretability for RUL prediction, but the complexity of machinery structures makes modeling difficult, which limits the versatility of these methods 5. Data-driven methods predict RUL by analyzing observed data. Despite their lower interpretability, they are gaining more attention due to their reduced reliance on expert knowledge 6. The hybrid model incorporates the advantages of the aforementioned two approaches but is limited by its dependence on expert knowledge and challenges in integrating data with models 7.

As the technologies of sensor and information processing advance rapidly, the monitoring data obtained from the industry are getting more abundant and comprehensive, laying a solid foundation for data-driven development. Simultaneously, owing to the rapid growth of deep learning (DL), the strong nonlinear mapping capabilities demonstrated by DL have opened new research avenues for RUL prediction. Many DL-based RUL prediction methods have been developed; such as deep belief networks (DBN) 8, convolutional neural networks (CNN) 9, and long short-term memory (LSTM) networks 10. The multi-objective deep belief network (MODBN) 8 integrates multiple trained DBNs into a single model to improve prediction accuracy and variety. Babu et al. 11 utilized the local perception and parameter-sharing capabilities of CNN for RUL prediction. Miao et al. 12 proposed a dual-task LSTM network to predict the RUL of turbine engines using monitoring signals from multiple sensors. Liu et al. 13 used an encoder-decoder structure that combined bidirectional long short-term memory (BiLSTM) networks and CNN to capture long-term dependencies and key local characteristics in the data for RUL prediction. Shao et al. 14 innovatively proposed a data-enhanced prediction method using artificial intelligence technology to supplement incomplete and missing variables in industrial equipment, followed by accurate prediction of the RUL of equipment using dynamic Bayesian networks and LSTM. The improved multi-stage long short-term memory network with clustering (ILSTMC) 15 integrated the clustering algorithm into the LSTM network for RUL prediction, which fully utilizes the advantages of both techniques. The bidirectional gated recurrent unit with the temporal self-attention mechanism (BiGRU-TSAM) 16 achieved better RUL prediction results by assigning larger weights to key temporal steps and characteristics.

However, many DL-based methods predominantly focus on modeling the temporal dependencies or using the grid as a model structure. This singular focus on temporal dependencies often neglects the spatial location information of sensors. While incorporating grid-structured data can provide some spatial information, this method is usually applied to regular data with constant neighboring nodes, which may not effectively capture the non-Euclidean spatial dependencies present in the data.

Graph convolutional networks (GCNs), currently the most popular method for processing non-Euclidean data, are extensively used for data expressed as graphs. For instance, Wang et al. 17 combined graph convolution with CNN to tackle the challenge of fetal head detection. Shin et al. 18 developed a progressive graph adaptive to data and combined it with dilated causal convolution to capture information in traffic flow data. Liang et al. 19 combined the graph attention network (GAT) and deep adaptive transformer to achieve accurate predictions by extracting temporal and spatial features from sensor data. As highlighted in 20, the effective extraction of spatio-temporal features from data is crucial for PHM.

Although research on extracting spatio-temporal features from non-Euclidean data using graph neural networks has achieved certain results, most studies overlook two key issues. The first issue, while constructing the graph structure, is the complex dependencies among sensors, making it challenging to establish physical variable connections among sensors. In the RUL prediction task, there is a lack of data, such as road distances in the transportation field, to use as a reference for calculating the weights of the adjacency matrix. Hierarchical attention graph convolutional network (HAGCN) 21, gated graph convolutional network (GGCN) 22, and graph convolutional attention network with temporal convolution-aware nested residual connections (GCN-TCNR) 23 construct graphs based on similarity, identifying adjacent nodes by calculating the cosine similarity of sensor data to form the adjacency matrix. However, this method risks neglecting the complex dynamic relationships among sensors. For example, if the time series signals collected by a fan speed measurement sensor and a temperature measurement sensor are considered as two sets of vectors, the cosine similarity would indicate minimal dependency if these vectors point in opposite directions, which does not accurately reflect real-world conditions. Additionally

to filter out insignificant connections, constructing graphs using cosine similarity requires choosing an appropriate threshold with considerable impact. Last but not least, this method has a limited field of view and risks ignoring some global information. Li T. et al. 24 summarized three common methods for constructing graph structures: KNNGraph, RadiusGraph, and PathGraph, with RadiusGraph using cosine similarity. KNNGraph reflects the local similarity among nodes by considering each node's nearest neighbors, and the k-value choice significantly affects the graph's quality. Moreover, calculating k-nearest neighbors becomes computationally intensive with large datasets. PathGraph directly connects the data's original time series, making it simple to construct. However, this method excessively relies on the order of data points and may overlook non-continuous but similarly patterned data, such as common periodic data. Ma et al. 25 proposed the adaptive graph convolutional transformer encoder (AGCTE) method, which does not require prior knowledge and adaptively learns dependencies among nodes by representing nodes as learnable embedding vectors. However, this static graph structure may fail to effectively capture spatial features well when node dependencies change over time. The unresolved challenge of dynamically capturing dependencies among sensors without relying on prior knowledge remains a critical issue.

The second issue concerns the potential dependencies among sensors across various time periods. In RUL prediction, the physical quantities measured by different sensors may differ, and some quantities, like temperature, may exhibit a certain delay. For instance, the temperature data recorded at a certain moment might strongly correlate with the vibration data from the preceding moment. However, most methods that consider sensor spatial information currently focus only on the relationships among different sensors at a single time and the same sensor at various times. For example, the adaptive spatio-temporal graph convolutional neural network with metric (ASTGCNN-Metric) uses generalized Mahalanobis distance and Gaussian kernel functions to adaptively learn dependencies among nodes 26, considering only node dependencies at the same time. As a result, dependencies among delayed data and other sensor data might be ignored. Deeply exploring relationships among sensors at different times and constructing cross-temporal sensor networks may enhance RUL prediction accuracy.

To address these issues, this article presents a cross-temporal dynamic graph convolutional network (CTDGCN) that aggregates information through spatio-temporal graph convolution adopts a dynamic graph structure, and constructs a cross-temporal sensor network, effectively capturing the data's spatio-temporal features and enhancing the accuracy of RUL prediction. The contributions of this article can be summarized as follows:

(1) A CTDGCN framework for RUL prediction is proposed, predicting simple components in the input and remodeling complex components through feedback, establishing spatio-temporal dependencies between sensors.

(2) A dynamic graph generation method is designed, integrating time embeddings, node embeddings, and input sensor signals to construct the dynamic graph. This method extracts dynamic spatial features from the input signals without requiring prior knowledge.

(3) A cross-temporal graph convolutional module (CTGCM) is proposed to construct a dependency network among sensors at different times. By combining the decay adjustment matrix and the power decay matrix, the cross-temporal sensor graph structure is modified, and the dependency information among sensors is aggregated in the cross-temporal pooling layer, making it easier for the model to learn long-term decay relationships.

The remaining contents are organized as follows: Section 2 introduces the definition of RUL prediction and the proposed CTDGCN model framework. Section 3 discusses the effectiveness of CTDGCN in RUL prediction through two case studies. Section 4 analyzes the model. Section 5 presents conclusions and prospects for future research.

2. Methodology

2.1. Preliminary

GCN is a network designed to handle graph-structured data. Unlike grid and sequence structures, graph structures are composed of nodes and edges, with edges describing the relationships among nodes. This can be represented by:

$$G = (V, \mathcal{E}, A), \tag{1}$$

where V represents the set of nodes, $\mathcal{E}$ represents the set of edges, and $A \in R^{N\times N}$ is the adjacency matrix. The elements in A indicate the relationships among nodes.

In RUL prediction, sensors are represented as different nodes, with edges between nodes whose weights indicate the strength of the relationships between different sensors. The sensor signal $X_t \in R^{N\times d}$ represents the signals collected by N sensors at time t, where d is the number of features of the collected signals. Predict RUL y_t at time t using the signals $X_{t-T+1:\,t} = (X_{t-T+1}, \ldots, X_t) \in R^{T\times N\times d}$ collected from all sensors within the time period T. Thus, the RUL prediction problem can be viewed as learning the mapping function f from the input graph structure G and the feature matrix $X_{t-T+1:\,t}$ to predict y_t, as shown in the following equation:

$$y_t = f(G;\, X_{t-T+1:\,t}). \tag{2}$$

GCN can be considered as the first-order approximation of spectral graph convolution 27. Its model can be expressed by the following equation:

$$Z = \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) X\Theta + b, \tag{3}$$

where I_N is the identity matrix, $D = diag(\sum_j A_{ij})$ is the degree matrix, A represents the graph structure, and it is a positive semi-definite matrix. $X \in R^{N\times d}$ is the feature matrix, $\Theta \in R^{d\times h}$ is the parameter matrix, and $b \in R^h$ is the bias vector. The output is $Z \in R^{N\times h}$.

2.2. Proposed CTDGCN method framework

This article proposes a GCN-based cross-temporal dynamic graph architecture for RUL prediction, consisting of a dynamic graph generation module and a cross-temporal spatio-temporal encoder (CTST), as shown in Figure 1. For RUL prediction, the collected multi-sensor data are first divided using the sliding window technique. Time and spatial embeddings are then added to the data to initially obtain the dynamic graph embeddings. The data containing spatio-temporal embedding information are fed into the CTST encoder. The RUL prediction results are generated through the output prediction channel, and data that are difficult to fit are extracted through the feedback prediction channel. This extracted data is used as the input for the next layer of the encoder, and the output of the next layer is used to refine the prediction results, as shown in Figure 1 (a).

2.3. Dynamic spatio-temporal graph generation

To fully extract spatio-temporal features from multi-sensor time series data, time position encoding and spatial node encoding are combined. This guarantees that the embedded information includes both local time information and node spatial information. The dynamic adjacency matrix captures varying node dependencies at different moments. To account for the local temporal position of the data and ensure the directionality of the time series, the position encoding method from 28 is used. This involves adding a positional encoding of length T to the input time series $X_{t-T+1:\,t}$. The formulation is as follows:

$$E^T_{(pos,2i)} = \sin\left(pos/L^{2i/f}\right), \tag{4}$$

$$E^T_{(pos,2i+1)} = \cos\left(pos/L^{2i/f}\right), \tag{5}$$

where pos represents the position encoding, f denotes the feature dimension, i denotes the i-th feature dimension, L represents the largest geometric progression, and $E^T \in R^{T\times N\times f}$ represents the time embedding.

In addition to time information, the spatial information of sensors also plays an important role in feature extraction. Spatial information is not limited to the distance between sensors but also includes dependency relationships between physical quantities. However, in many cases, the spatial information between sensors is not clear. Therefore, learnable parameters $E^N \in R^{N\times f}$ are used as node embeddings. The local time embedding is multiplied by the spatial node embedding to obtain the spatio-temporal embedding $E^{T,N}_t \in R^{N\times f}$:

$$E^{T,N}_t = E^T_t \odot E^N. \tag{6}$$

The signals collected by the sensors are passed through an MLP layer to extract dynamic information 29. The extracted information is then multiplied by $E^{T,N}_t$ and activated by the tanh function to obtain the dynamic graph embedding, which can be represented as follows:

$$E_t = \tanh\left(E^{T,N}_t \odot \mathrm{MLP}(X_t)\right). \tag{7}$$

Similar to the way graphs are built using cosine similarity, spatial information is learned by multiplying E_t with its transpose, E^{T}_t. To satisfy the requirements of Chebyshev polynomials, the adjacency matrix at time t in the dynamic adjacency matrix can be represented as $A_t = \mathrm{ReLU}(E_t E^{\mathrm{T}}_t)$. Therefore, Z at time t in equation (3) can be expressed as follows:

$$Z_t = \left(I_N + D^{-\frac{1}{2}}\left(\mathrm{ReLU}(E_t E_t^{\mathrm{T}})\right)D^{-\frac{1}{2}}\right)X_t\Theta + b, \qquad (8)$$

where $\Theta \in R^{N\times d\times h}$ is the weight matrix. To incorporate node features into the weights and enhance the node learning capability, the weight matrix is decomposed into the node embedding $E^N \in R^{N\times f}$ and the weight learning matrix $W^\Theta \in R^{f\times d\times h}$. Similarly, the bias matrix $b \in R^{N\times h}$ is obtained by multiplying the node embedding E^N with the bias learning matrix $b^\Theta \in R^{f\times h}$. Therefore, the GCN can be expressed as follows:

$$Z_t = \left(I_N + D^{-\frac{1}{2}}\left(\mathrm{ReLU}(E_t E_t^{\mathrm{T}})\right)D^{-\frac{1}{2}}\right)X_t E^N W^\Theta + E^N b^\Theta, \qquad (9)$$

where $X_t \in R^{N\times d}$ signifies the input to the network, while $Z_t \in R^{N\times h}$ signifies the output of the network.

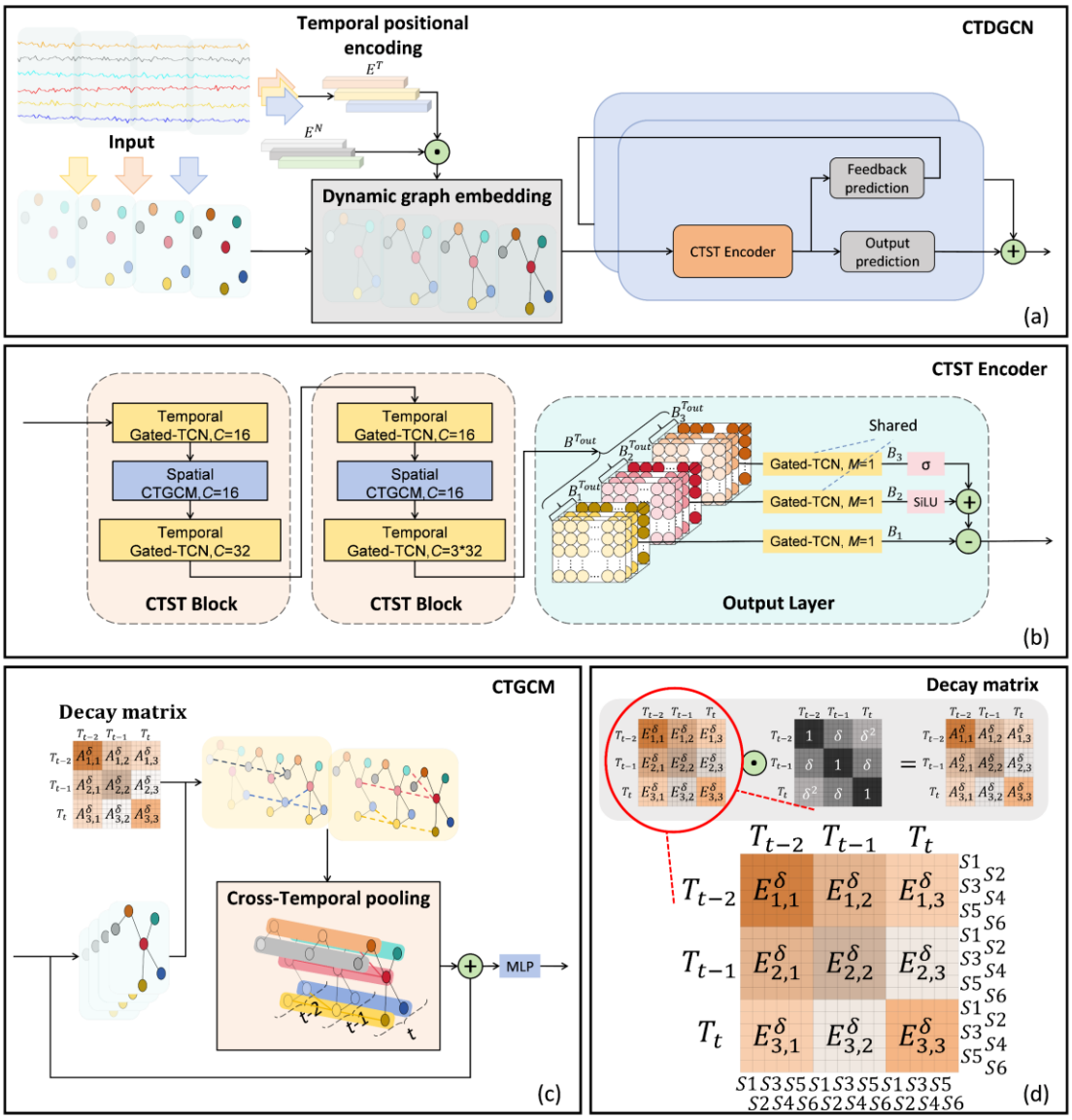


Figure 1. The architecture and modules of CTDGCN. (a) The architecture of CTDGCN; (b) The modules of CTST encoder; (c) The modules of CTGCM; (d) Decay matrix.

2.4. Cross-temporal spatio-temporal encoder

The model constructs the CTST encoder by merging gated temporal convolution with CTGCM to more effectively capture the spatio-temporal dependencies among sensor data. The CTST encoder contains two CTST blocks with one output layer, as shown in Figure 1(b). To extract temporal features, 1D causal convolution and a gating mechanism are employed in the temporal layer (TL) 30. For convenience, the time dimension is zero-padded at the end. The temporal characteristics of the sensor signal, when processed through TL, can be formulated as follows:

$$(U_{t-T+1}, \dots, U_t) = \mathrm{TL}(X_{t-T+1}, \dots, X_t), \qquad (10)$$

where $U_t \in R^{N\times f}$ represents the advanced expression of sensor signals at time t. The n-th row in U_t corresponds to the advanced embedding of the n-th sensor at time t after processing through the TL.

To capture the spatial correlations of nodes over multiple time periods, this article designs the CTGCM. This module leverages U_t from different time points as input to produce spatial embeddings, as depicted below:

$$S_t = \mathrm{CTGCM}(U_t, A_t), \tag{11}$$

where A_t denotes the adjacency matrix at time t, the CTST block is built with a TL-CTGCM-TL structure. The encoder's output channel number is C_{eout}. By stacking multiple blocks and expanding the feature dimension channels to $3C_{eout}$ in the final TL, we obtain the spatio-temporal embedding sequence $B_t \in R^{N\times 3C_{eout}}$. To aggregate embeddings from different times, we obtain $B^{T_{out}} \in R^{T_{out}\times N\times 3C_{eout}}$, where T_{out} represents the embedding sequence length after the CTST block. The output layer divides $B^{T_{out}}$ into three parts: $B_1^{T_{out}}$, $B_2^{T_{out}}$, and $B_3^{T_{out}}$. Each part is processed via 1D causal convolutions and gating mechanisms, reducing the temporal dimension T_{out} to 1, which results in B_1, B_2, and B_3. Note that $B_2^{T_{out}}$ and $B_3^{T_{out}}$ share convolution parameters.

Finally, the l-th layer encoder output is represented as:

$$H^l = B_1 - \left(\sigma(B_2) + \mathrm{SiLU}(B_3)\right), \tag{12}$$

where $\sigma(\cdot)$ and $\mathrm{SiLU}(\cdot)$ denote the Sigmoid and SiLU activation functions, respectively. These functions are utilized to extract lower-value features from the data and subtract these elements.

To extract feature signals that a single encoder cannot detect, a feedback structure is implemented for multi-layer prediction. The encoder's output is processed using a simple CNN operation in both the output prediction and feedback prediction channels, as described below:

$$\hat{y}^l = \mathrm{CNN}_{\mathrm{output}}(H^l), \tag{13}$$

$$X^{l+1} = \mathrm{CNN}_{\mathrm{feedback}}(H^l), \tag{14}$$

where l represents the layer number of the encoder, the output prediction $\hat{y}^l$ is a constant that signifies the predicted RUL value for that layer. $X^{l+1} \in R^{T\times N\times d}$ is used as input to the next layer, and the dropout rate for the next layer is reduced to half of the preceding layer. By re-extracting complex features through the feedback structure, the RUL prediction is further refined. The final predicted RUL value $\hat{y}$ is calculated by summing $\sum_{l=1}^{L}\hat{y}^l$, where L denotes the overall layer of the encoder.

2.5. Cross-temporal graph convolutional module

To handle the dependencies among sensors across different periods, the CTGCM in CTDGCN is designed to create the cross-temporal adjacency matrix, as shown in Figure 1(c). The cross-temporal graph structure is constructed using the dynamic embeddings E_t from the sensors. These embeddings E_t are then concatenated by sliding through adjacent timestamps to obtain $E_t^M \in R^{3N\times f}$, where M denotes the temporal span size. Similarly, X_t^M is utilized as input to CTGCM. The cross-temporal dependencies in the dynamic adjacency matrix at time t are represented as $A_t^M = E_t^M (E_t^M)^{\mathrm{T}}$. Intuitively, the correlation among sensors at closer times should be stronger than that among sensors at more distant times 31.

Therefore, a power decay matrix A_d^δ with a decay factor of δ is used to attenuate the weights, and in this paper, δ=0.7 is chosen as the model decay parameter. The decay rates in various studies can differ significantly. To address this, an adaptive decay adjustment matrix is introduced to fine-tune the decay effects on timestamps adaptively, thereby minimizing the model's sensitivity to decay coefficients. The adaptive decay adjustment matrix and the power decay matrix have the same form, as shown in Figure 1(d). The adaptive decay adjustment matrix can be expressed as:

$$A_a^\delta = \left\{E_\delta {E_\delta}^{\mathrm{T}}\right\}_{i,j=0}^{N}, \tag{15}$$

where $E_\delta \in R^{M\times 1}$ is composed of learnable parameters, and $A_a^\delta \in R^{MN\times MN}$ is derived by expanding $E_\delta {E_\delta}^{\mathrm{T}}$ in both rows and columns N times. At the same time, a single decay factor is used by the sensor for adjustment, which is used to control the specific decay effect. Figure 1(d) shows the decay relationships of six sensors at three time points. Each row represents the dependencies of a sensor at the corresponding time point with the sensors at time points $t-2$, $t-1$, and t.

The cross-time adjacency matrix is obtained by element-wise multiplication of A_a^δ and A_d^δ, and it is constrained within the range [0,1] using the softmax function. This can be expressed as:

$$A^\delta = \mathrm{softmax}\left(A_a^\delta \odot A_d^\delta\right), \tag{16}$$

where $A^\delta \in R^{MN\times MN}$ represents the cross-temporal adjacency matrix. Consequently, equation (9) can be rewritten as:

$$Z_t^M = \left(I_N + D^{-\frac{1}{2}}\left(\mathrm{ReLU}\left(A_t^M A^\delta\right)\right)D^{-\frac{1}{2}}\right)X_t^M \Theta^M + b^M = \left(\mathrm{I_N} + D^{-\frac{1}{2}}\left(\mathrm{ReLU}\left(A_t^M A^\delta\right)\right)D^{-\frac{1}{2}}\right)\mathrm{X_t^M} E^M W^\Theta + E^M b^\Theta, \quad (17)$$

where $\Theta^M \in R^{MN\times f\times h}$ denotes the weight matrix of the cross-temporal dynamic graph, $E^M \in R^{MN\times d}$ is obtained from the embeddings E^N of the M nodes, and the output is represented as $Z_t^M \in R^{MN\times h}$.

By constructing a cross-temporal dynamic graph, dependencies among sensors at different times are extracted. The CTGCM output serves as input to the next module for extracting temporal features. Each time slice in the cross-temporal graph contains information from that time and the previous $(M-1)$ moments, which may lead to redundant and repetitive information when extracting temporal features, resulting in additional computational load. To address this issue, an average pooling strategy is used to pool the M identical sensors processed by the cross-temporal graph, obtaining the output at time t. Subsequently, the output information is transformed and represented through the MLP layer.

3. Experimental analysis

To validate the effectiveness of CTDGCN, this section compares its performance with existing RUL prediction models by two case studies. CTDGCN is implemented using Python 3.8 and the PyTorch 1.13 framework. Additionally, the case studies are executed on a computer equipped with an Intel i9-12900K CPU and a GTX 4090 GPU.

3.1. Case study I: RUL prediction of CMAPSS dataset

3.1.1. Description of the CMAPSS dataset

In this case study, experiments are conducted using the aircraft turbofan engine full lifecycle dataset available from the NASA database. This dataset is acquired via the Commercial Modular Aero-Propulsion System Simulation (CMAPSS) 32. The turbofan engine's major parts consist of the fan, low-pressure compressor (LPC), combustor, high-pressure compressor (HPC), low-pressure rotor (N1), high-pressure rotor (N2), low-pressure turbine (LPT), high-pressure turbine (HPT), and nozzle. The CMAPSS dataset consists of four sub-datasets: FD001, FD002, FD003, and FD004, as shown in Table 1.

Table 1. Details of the CMAPSS dataset.

CMAPSS dataset	FD001	FD002	FD003	FD004
Training engine units	100	260	100	249
Testing engine units	100	259	100	248
Operating conditions	1	6	1	6
Fault modes	1	1	2	2
Maximum cycle number	362	378	525	543
Minimum cycle number	128	128	145	128

Every subset includes a training set and a testing set, and each training and testing set contains 26 data columns. These columns denote the engine number, cycle number, three operational settings, and measurements from 21 sensors. During the experiment, the engine is running with negligible wear in the early stages of operation and is considered to be normally operating. As the engine runs, faults accumulate, reducing the RUL until the engine fails. In the training set, the dataset includes complete run-to-failure cycles. In the testing set, the data ends at a point before the engine fails. The prediction task is to use the engine's degradation from the measured data, and the results are validated using the real RUL of the engine.

3.1.2. Data preprocessing

The values recorded by different sensors have different distributions. The raw time-series signals of the 21 sensors in the FD001 are plotted, as shown in Figure 2. Some sensor values are constant, making it meaningless to include them in the prediction task. Therefore, to ensure that valuable information is extracted, the operating data of sensors numbered 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20, and 21 are selected for the experiment.

In the initial stage, the equipment is relatively stable with a low degree of component damage. Therefore, a maximum RUL is defined. If the RUL of the data exceeds this value, it is set to the maximum value. After operating for a certain period, the degree of damage accumulates, and the RUL gradually decreases. The RUL threshold is established as $R_{max} = 125$, and the RUL is described by a segmented linear degeneracy

function, as shown below:

$$RUL = \begin{cases} R_{max}, R \geq R_{max} \\ R, R < R_{max} \end{cases}, \tag{18}$$

Due to the time-series value ranges measured from various sensors, directly inputting them into the neural network may result in the network failing to converge, thus not achieving the expected prediction results. Therefore, data processing is required. In this paper, the data is normalized to [0, 1] using the Z-score. Since FD002 and FD004 have different operating conditions, K-means clustering is performed first, followed by normalization for each cluster separately 21. The overall process can be represented as follows.

$$x_i^{norm} = \frac{x_i^c - \mu_i^c}{\delta_i^c}, \tag{19}$$

where x_i^c denotes the i-th sensor's data collected under operating condition c; x_i^{norm} denotes the i-th sensor's normalized value; μ_i^c and δ_i^c represent the i-th sensor's data collected under operating condition c's mean value and standard deviation, respectively.

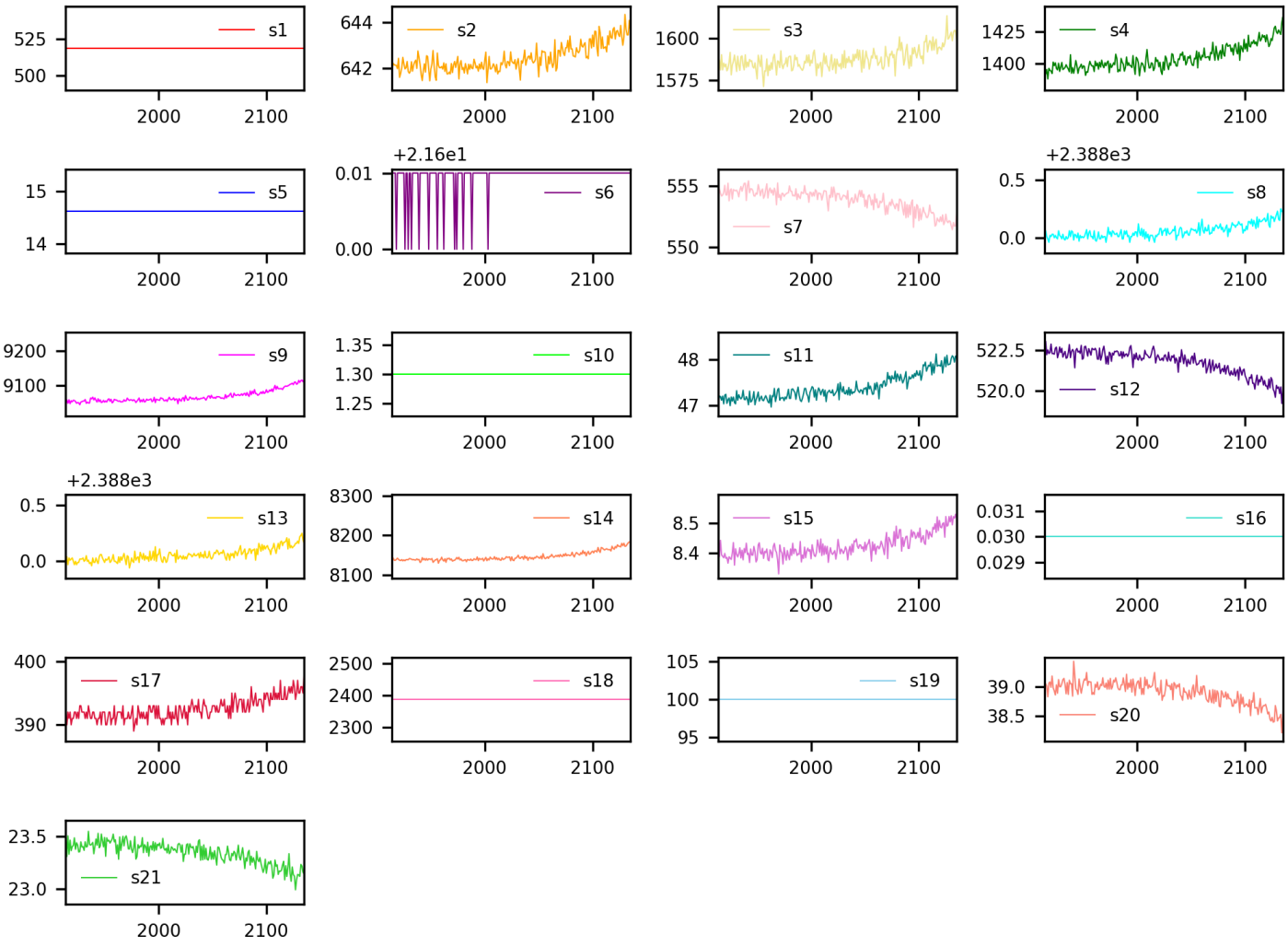


Figure 2. Raw time series data for turbofan engine #10 in FD001.

3.1.3. Evaluation metrics

To assess the model's performance, this paper employs two indicators: the root mean square error (RMSE) and the score function (SF). RMSE is extensively utilized to evaluate model performance in regression tasks. It measures the deviation between the predicted RUL and the real RUL by calculating the RMSE. The formula is defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{P}\sum_{i=1}^{P}\left(\hat{Y}_i - Y_i\right)^2}, \tag{20}$$

where $\hat{Y}_i$ and Y_i denote the i-th sample's RUL predicted and real values, respectively, and P denotes the total number of samples.

In industrial settings, late-stage predictions of RUL can result in more severe consequences than early-stage predictions.

Therefore, the SF is frequently used as an evaluation metric. It imposes greater penalties on late-stage predictions, thus addressing the limitations of RMSE in this regard. The definition is as follows:

$$\mathrm{SF} = \sum_{1}^{P} \mathrm{SF}_i ,$$

$$\text{with } \mathrm{SF}_i = \begin{cases} e^{-\frac{\hat{Y}_i - Y_i}{13}} - 1, \text{for } \hat{Y}_i \le Y_i \\ e^{\frac{\hat{Y}_i - Y_i}{10}} - 1, \text{for } \hat{Y}_i \ge Y_i \end{cases}, \tag{21}$$

In predicting RUL using deep learning models, the RMSE value changes linearly with the error, whereas the SF imposes greater penalties on late-stage predictions, as illustrated in Figure 3. Both RMSE and SF have clear physical significance for evaluating prediction performance. Consequently, this study uses these two indicators to evaluate model performance.

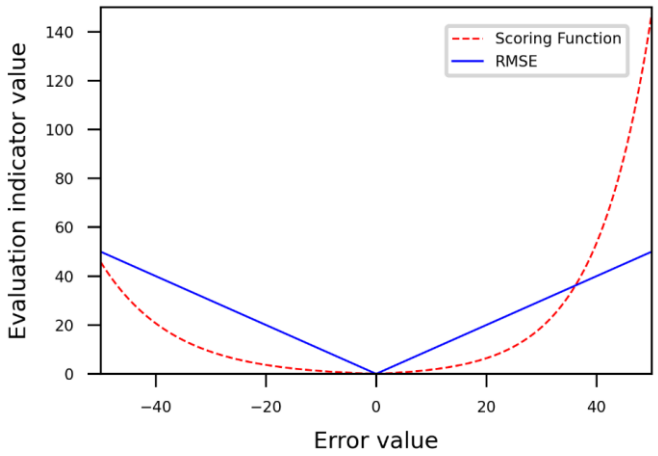


Figure 3. Comparison of RMSE and SF.

3.1.4. Experimental results

To determine the optimal model parameters, a parameter search is conducted for the best performance. The optimizer used is Adam, with a batch size of 256. The total training consists of 50 epochs; the window length is 30, the dropout rate is 0.2, and the initial learning rate is 0.003. The learning rate is decayed by a factor of 0.1 at the 25th and 40th epochs.

Table 2 compares the prediction results of the CTDGCN model and the sixteen methods. Traditional models that only model time are not well-suited for RUL prediction. While some non-graph models that consider both temporal and spatial features show improved performance, their ability to extract non-Euclidean data is limited, resulting in relatively inferior performance compared to GCN-based methods. STGCN, ASTGCNN-Metric, HAGCN, GAT-EdgePool, GGCN, AGCTE, and the proposed model can capture temporal characteristics and non-Euclidean space dependencies by combining GCN.

This is especially evident in the complex subsets FD002 and FD004. However, STGCN requires the definition of an adjacency matrix. In RUL prediction, it is challenging to find a physical quantity that is similar to the connectivity and distance relationships between nodes in a traffic flow dataset. Consequently, this paper adopts the approach of calculating cosine similarity between sensors using HAGCN, GGCN, and AGCTE to determine the graph structure, thereby helping STGCN establish node dependencies.

Table 2. Experimental results comparison on CMAPSS test dataset.

Methods	FD001		FD002		FD003		FD004	
	RMSE	SF	RMSE	SF	RMSE	SF	RMSE	SF
DCNN 11	15.65	449	24	10300	18.84	460	28.74	6780
STGCN 30	14.92	343.20	15.78	1315.68	16	991.62	18.73	2265.41
DAGN 34	16.11	595	16.43	1242	18.05	1216	19.04	2321
ASTGCNN-Metirc 26	15.06	459	17.34	1486.66	15.34	599	19.77	2517.2
BiLSTM-MSCNN 35	12.75	281	22.46	5170	11.35	278	24.1	4790
HAGCN 21	13.1	263	14.92	1086	13.46	327	15.74	1218.6
GAT-EdgePool 24	13.53	244.6	16.86	1526.13	14.55	319.51	16.23	1497.03
BiGRU-TSAM 16	12.56	213.35	18.94	2264.13	12.45	232.86	20.47	3610.34
IDMFFN 36	12.18	204.69	19.17	1819.42	11.89	205.54	21.72	3338.84
GGCN 22	11.82	186.70	17.24	1493.7	12.21	245.19	17.36	1371.5
MSDCNN-LSTM 37	12.96	256.59	18.70	1873.86	11.78	211.99	21.57	2699.34
STRUL 38	12.85	224	19.24	1950	13.74	252	22.34	3080
IMDSSN 39	12.14	206.11	17.4	1775.15	12.35	229.54	19.78	2852.81
AGCTE 25	12.46	259.37	13.7	833.41	12.95	372.44	15.83	1520.05
TATFA-Transformer 40	12.21	261.5	15.07	1359.7	11.23	210.21	18.81	2506.35
MHT 41	11.92	215.2	13.7	746.7	**10.63**	**150.5**	17.73	1572
CTDGCN	**11.34**	**180.93**	**13.27**	**728.19**	11.32	232.92	**14.26**	**938.73**

Compared to methods that require pre-defined graph structures, the approach proposed in this paper constructs graph structures without prior knowledge, autonomously learning the dependencies between nodes. Unlike HAGCN, GAT-EdgePool, and GGCN, which use cosine similarity to construct graphs, the proposed model uses a dynamic adjacency matrix to capture more comprehensive spatial dependencies between nodes, demonstrating a stronger ability to learn complex dependencies. In contrast to the adaptive graph construction methods of ASTGCNN-Metric and AGCTE, the CTDGCN architecture dynamically captures hidden spatial correlations between sensors over different time points and is simpler to implement. The comparison demonstrates that the CTDGCN framework obtains optimal performance on FD001, FD002, and FD004. On FD003, CTDGCN, TATFA-Transformer, and BiLSTM-MSCNN perform similarly, with all three trailing behind the MHT method. However, the performance of MHT, TATFA-Transformer, and BiLSTM-MSCNN is average on FD001. On the more complex subsets FD002 and FD004, the RMSE and SF of the CTDGCN model are significantly lower than those of other models. This indicates that CTDGCN has a strong ability to extract spatio-temporal characteristics from complex data, showing great potential for RUL prediction under complicated conditions.

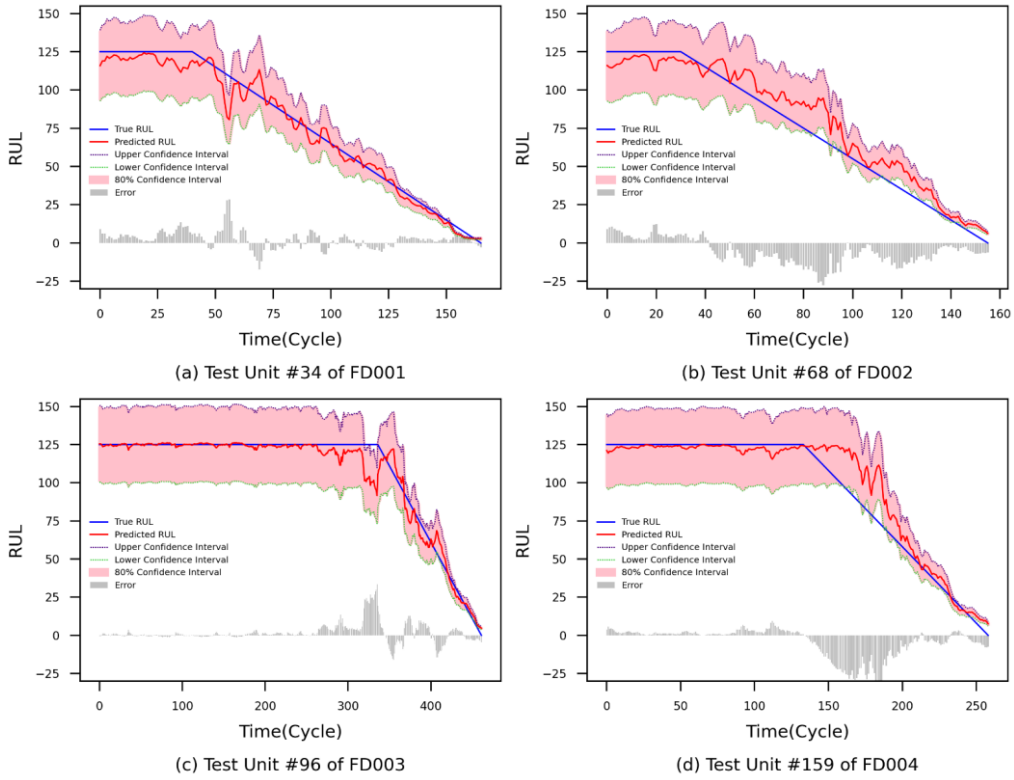


Figure 4. Example of RUL prediction on CMAPSS dataset.

Figure 4 presents prediction examples for the four subsets in the CMAPSS dataset. The information in the figure leads to the conclusion that, during the early stages when no failure is detected, the predicted RUL is near the maximum RUL. As time progresses, the real RUL degrades linearly. Despite some discrepancies, CTDGCN achieves high prediction accuracy, particularly near the failure point where the predicted values closely match the real RUL. Figure 5 compares the predicted RUL and real RUL of all engines in the test set, and visualizes the error between them. The prediction results are close to the actual values and agree with those in Table 2, which confirms that CTDGCN performs excellently in predicting RUL. Furthermore, a set of engine data was selected from each of the four sub-datasets, and the output results of the CTST encoder were processed using t-SNE dimensionality reduction technology, followed by visualization, as shown in Figure 6. It can be seen from the figure that early data shows a pronounced clustering trend, while in the degradation phase, the data becomes more concentrated. This indicates that the model successfully extracted degradation features from the data, thereby effectively ensuring the accuracy of RUL prediction.

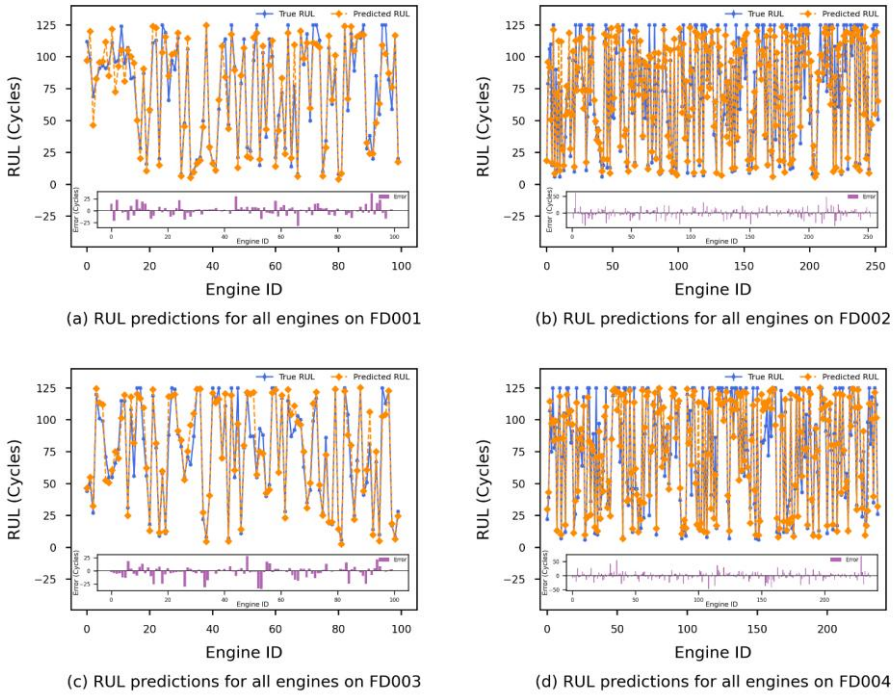


Figure 5. Comparison of predicted RUL and real RUL for all engines in the CMAPSS dataset.

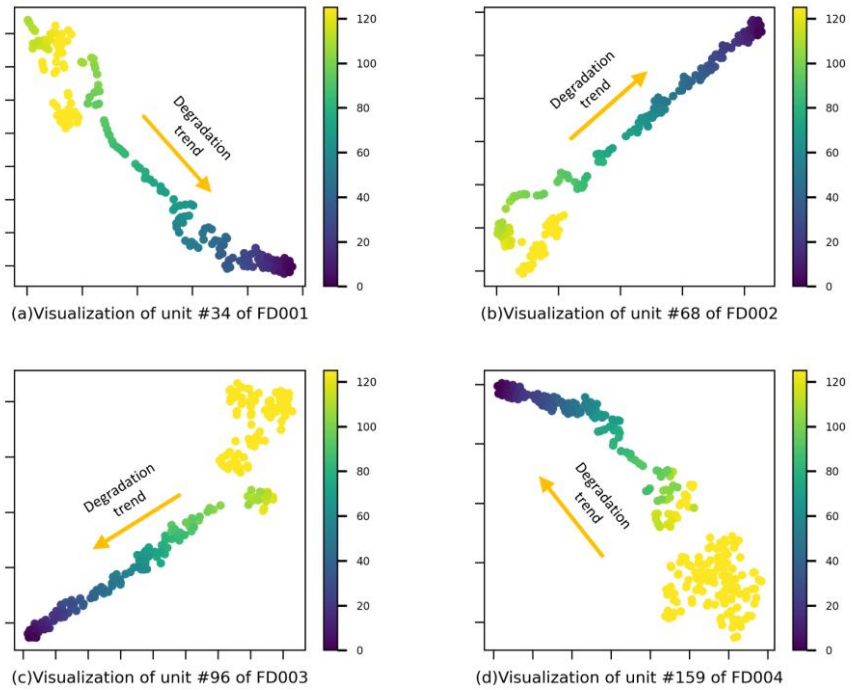


Figure 6. CMAPSS data set visualization.

3.2. Case study 2: RUL prediction of tool wear dataset

3.2.1. Tool wear dataset description

The tool wear dataset, provided by the PHM Society, documents a high-speed CNC machine tool operating at spindle speeds of up to 42,000 rpm 33. During the experiments, six cutters were independently used to machine stainless steel workpieces, with the flank wear of the cutters recorded as the label. The dataset comprises six subsets: C1, C2, C3, C4, C5, and C6, each containing 315 samples corresponding to 315 wear conditions. Degradation process data were collected from seven sensors that captured milling force and vibration signals in the X, Y, and Z directions, as well as the acoustic emission signal RMS values.

As only subsets C1, C4, and C6 include tool wear labels, this study utilized these three subsets for the experiments.

3.2.2. Data preprocessing

The original signal has an excessive sample rate, resulting in an excessive sample size. To address this, the original data is downsampled, and every sample is split into 30 groups, with the RMS value of each group calculated as the statistical characteristic. These values are then normalized using min-max normalization. In the test case, two sub-datasets are selected for training, and the remaining sub-dataset is utilized for testing, as shown in Table 3.

Table 3. Details of tool wear data set allocation.

Training set	Testing set	Sample of train	Sample of test
C1+C4	C6	306+278	238
C1+C6	C4	306+238	278
C4+C6	C1	278+238	306

The average wear of the three flutes is calculated as the tool's wear. The milling cutter RUL is defined as follows:

$$\mathrm{RUL}(t) = t_w - t, \tag{22}$$

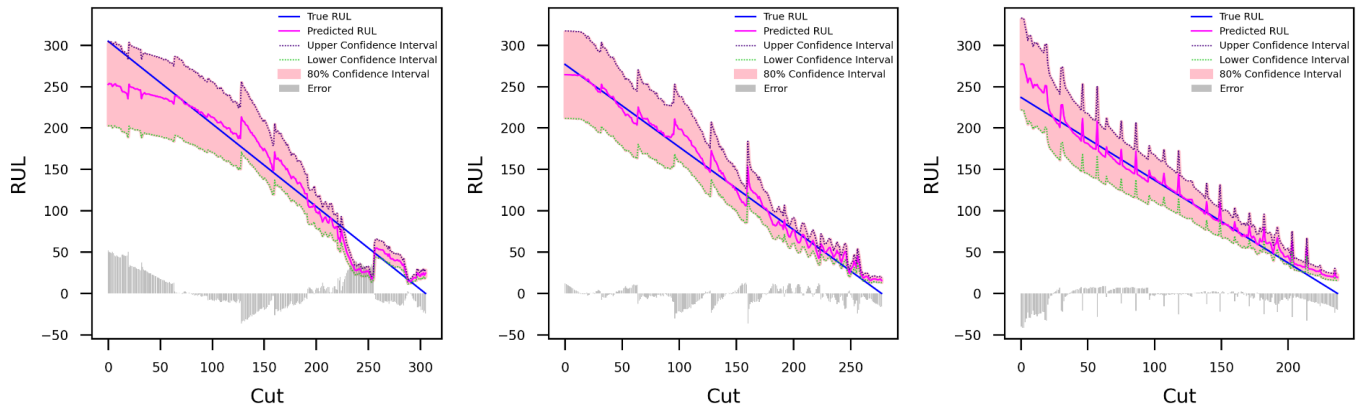
where t_w denotes tool wear time to failure, and t represents the current time. According to 21, the failure threshold t_w for the experiments is established at 0.16 mm.

3.2.3. Experimental results

For Case 2, the total training is 300 epochs, the initial learning rate is 0.02, and the learning rate is decayed by a factor of 0.1 at the 25th and 40th epochs. Other parameters are consistent with those in Case 1. A comparison with nine models is presented, with the results shown in Table 4. The proposed method demonstrates significant improvements over all existing models. Furthermore, Figure 7 illustrates the real RUL, the predicted RUL, and the error between them for the three tools in the experiment. It is evident that, in most cases, the predictions were accurate.

Table 4. Performance comparison of different models in tool RUL prediction.

Methods	C1		C4		C6	
	RMSE	SF	RMSE	SF	RMSE	SF
DCNN 11	37.4	11391.2	24.8	6147.2	41.1	361118.7
STGCN 30	30.98	5583.63	13.83	1038.99	33.57	30158.16
DLSTM 12	25.9	3065	14.5	815	30.2	12898
BLSTM 42	27.7	2981	12.8	623	33.7	19966
CNN-LSTM 43	44.7	46929.1	15.1	829.7	31.6	9673.1
D-CNN 44	37.3	9673	36.8	33614	33.5	266151
HAGCN 21	23.6	2278.3	11.2	589.6	15.6	1162.8
GAT-EdgePool 24	25.12	76729.95	19.27	8078.6	32.61	9000.55
STRUL 38	22.5	2912	11.9	**485**	13.1	1010
CTDGCN	**21.79**	**2153.14**	**10.63**	584.48	**11.49**	**612.77**



(a) RUL predictions for dataset C1 (b) RUL predictions for dataset C4 (c) RUL predictions for dataset C6

Figure 7. RUL prediction results for the tool wear dataset.

Although the proposed method's prediction SF for the C4 data subset is higher than STRUL, as shown in the C4 prediction in Figure 7, this may be due to fluctuations in the predictions. However, the prediction trend does not significantly deviate

from the real RUL decline pattern, and the degree of late-stage prediction is not severe. Considering the RMSE of CTDGCN on the C4 data subset and its performance on other datasets, this is deemed acceptable. There are some fluctuations in the C1 data subset; however, from the comparative experiments, it is evident that other methods also exhibit a significant decrease in accuracy on C1. This may be due to relatively less training data available when testing C1. From the above experimental results, it is clear that CTDGCN can provide precise prediction outcomes. By arranging reasonable maintenance timing and strategies based on the prediction results, it can better meet the requirements for equipment reliability and stability.

4. Model analysis

4.1. Effect of sliding window length on prediction

To explore the sliding window length effect on model performance, the optimal sliding window length in the range of {6, 12, 18, 24, 30, 42, 48, 54, 60} is searched across the four datasets.

The experimental results are illustrated in Figure 8. The model's performance progressively improves with larger window lengths. The reason is that a longer window can contain much greater information, allowing for better capture of degraded characteristics in the data by the model.

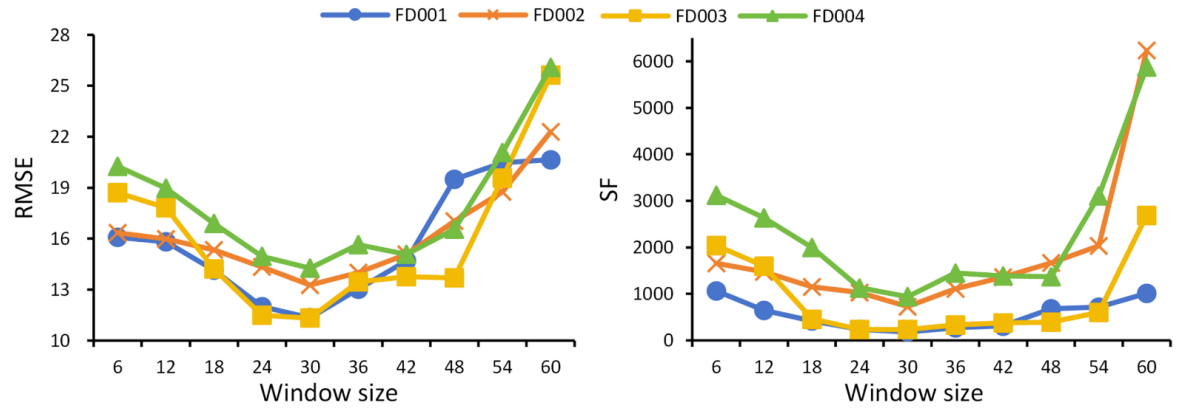
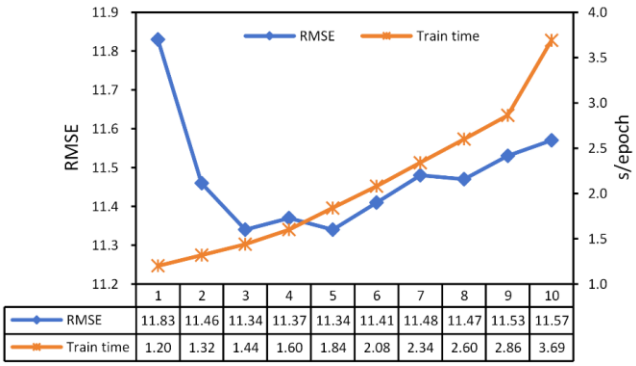


Figure 8. Impact of different windows on RUL prediction.

When the window length increases to around 30, CTDGCN achieves optimal performance. Further lengthening of the window does not considerably enhance model performance and may even cause a decline, likely due to increased computational burden and excessive complexity.

4.2. Impact of cross-temporal length on prediction

To explore the model's performance across different time spans, experiments are conducted on the FD001 with time spans ranging from 1 to 9, and the results are illustrated in Figure 9. The model achieves the best performance when the time span is 3. Both higher and lower time spans may not yield optimal results. Additionally, comparing the performance of a time span of 2 with that of 60 in the FD001 from Figure 8, it is clear that a time span of 2 performs significantly better. This is because merely extending the sliding window accumulates information along the time dimension while spanning time periods allows the model to learn dependencies among sensors at different times. The decay matrix establishes connections among these time periods, and the cross-temporal pooling layer summarizes the features, effectively exploring the relationships among sensors across different spatial and temporal dimensions.



	1	2	3	4	5	6	7	8	9	10
RMSE	11.83	11.46	11.34	11.37	11.34	11.41	11.48	11.47	11.53	11.57
Train time	1.20	1.32	1.44	1.60	1.84	2.08	2.34	2.60	2.86	3.69

Figure 9. Performance comparison across different time spans on FD001

4.3. Ablation study

For analyzing this model's improved parts contribution, this article compares four variants to validate the effectiveness of the improvements, as shown in Table 5. Model A removes the

temporal embedding and node embedding. Since there is no node embedding to adaptively generate the graph, the graph structure is constructed using cosine similarity. Model B replaces dynamic graph embeddings with a static graph to represent node dependencies. Model C eliminates the structure that spans different time periods, extracting spatio-temporal features from a single time span only. Model D removes the decay matrix. Model E represents the complete CTDGCN. All other parameter settings are the same as in the previous study, and the result is illustrated in Figure 10.

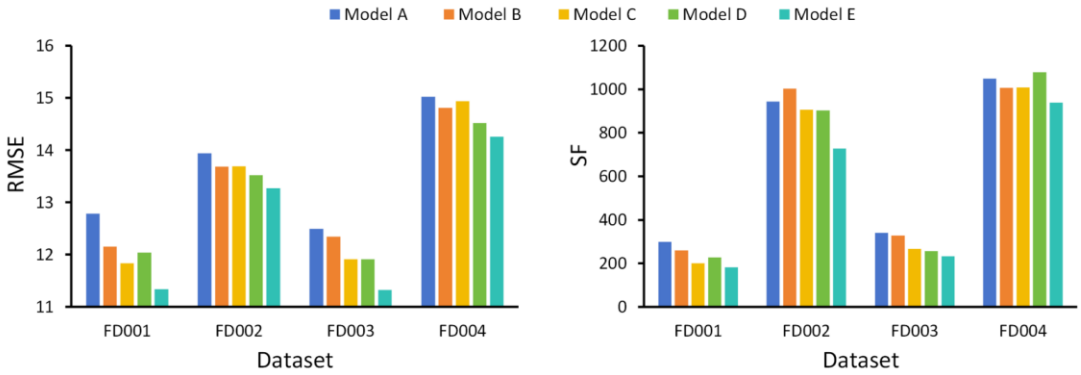


Figure 10. Experimental results of ablation study.

Table 5. Details of ablation experiment.

Model	Description
Model A	CTDGCN without embeddings
Model B	CTDGCN without dynamically constructing graphs
Model C	CTDGCN without cross-temporal structures
Model D	CTDGCN without decay matrices
Model E	CTDGCN

The experimental results reveal that Model A, which omits both time embedding and node adaptive embedding and relies solely on cosine similarity to construct the graph, inadequately captures the spatio-temporal characteristics in multi-sensor data, leading to poor performance. Model B, which uses a static adaptive graph, fails to capture the varying dependencies of sensors at different times. As a result, it cannot learn the dynamic spatial structure in the data, leading to limited improvements. A comparison between Model C and Model D shows that cross-temporal modeling without a decay matrix does not necessarily surpass models lacking such modeling, highlighting the importance of incorporating a decay matrix to enable trend learning. The experimental results indicate that CTDGCN, with its dynamic graph structure and cross-temporal modeling capabilities, achieves the best results, demonstrating the superiority of CTDGCN in RUL prediction.

4.4. Adjustment effect of adaptive decay adjustment matrix on power decay matrix

The ablation experiments indicate that the decay matrix significantly boosts the model's performance when forming connections across different time periods.

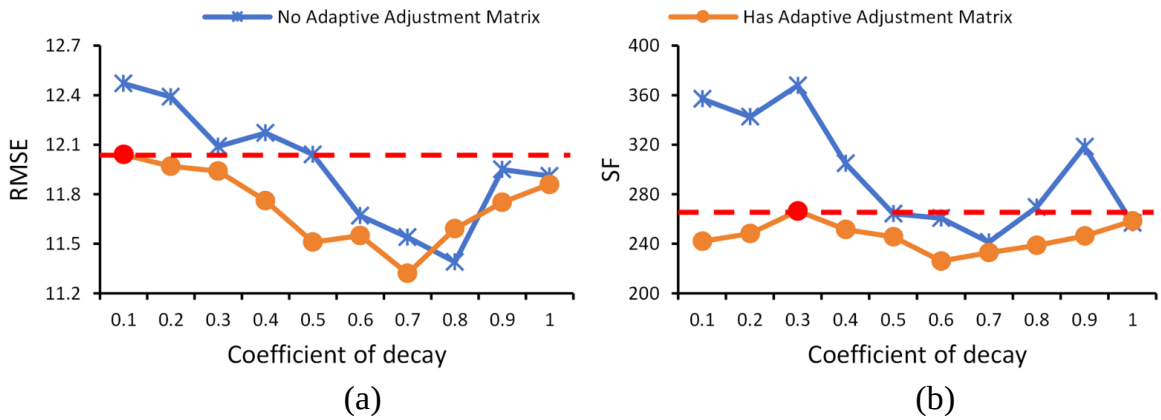


Figure 11. Comparison of the effects with and without the decay adjustment matrix on FD003

To assess how the decay adjustment matrix influences the performance of the model, the study compares models with and without the decay adjustment matrix under power decay matrices with varying decay coefficients, as depicted in Figure 11. It is evident that without a decay adjustment matrix, the appropriateness of the selected decay coefficient leads to substantial performance variability. Introducing the decay adjustment matrix reduces the model's RMSE to below 12.04 and SF to below 241.86. This reduces sensitivity to decay coefficient settings and simplifies the selection of an optimal coefficient. Additionally, we observe that the model incorporating the attenuation adjustment matrix achieves the best results in RMSE with a decay coefficient of 0.7. Although the SF result is suboptimal, it is only slightly worse than the best performance. Therefore, the model ultimately employs 0.7 as the coefficient for the attenuation matrix.

4.5. Analysis of dynamic graph generation

The node embedding dimension is a critical parameter for obtaining a dynamic graph. It affects both the parameter diversity of the generated graph structure and the learning capacity of the dynamic graph. Figure 12 illustrates the model's performance at various embedding sizes on FD002. A small embedding dimension limits the model's learning ability, while a larger dimension captures more information but increases parameter counts and the risk of overfitting. Experimental results demonstrate that an embedding length of 16 provides optimal performance for CTDGCN on this dataset.

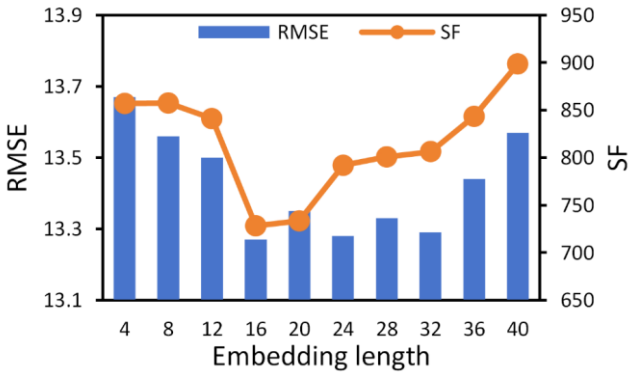
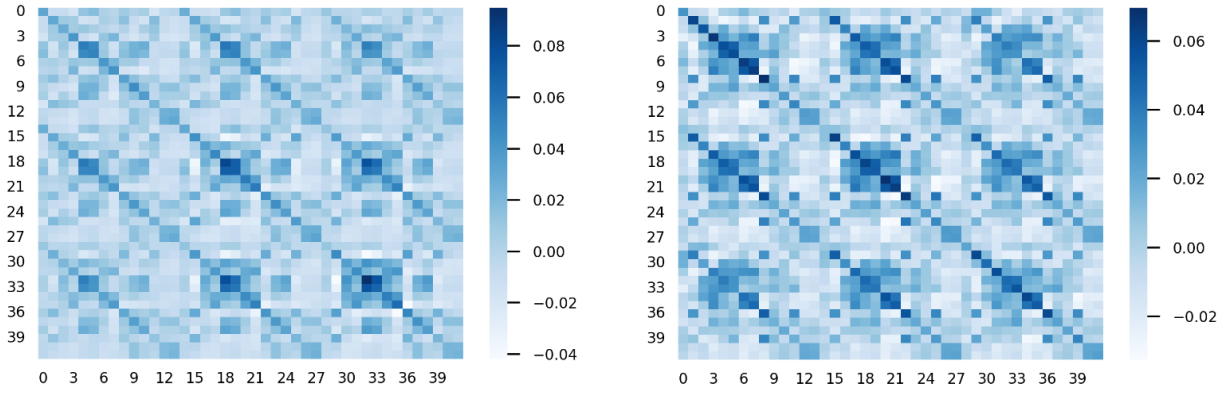


Figure 12. Performance of CTDGCN with different embedding dimensions on dataset FD002.

The learned adjacency matrices for the same batch at two different time periods are visualized in Figure 13, where darker colors represent stronger connections. It can be observed that even at different time periods, sensors generally exhibit strong self-dependency, which aligns with the experimental expectations.



(a) (b)

Figure 13. Adjacency matrices for two time periods in the same batch.

Moreover, the adjacency matrices display varying degrees of attention across different time periods. This observation is consistent with the design concept of the decay matrix, thereby validating its underlying mechanism. Finally, the sensor dependencies across different time periods are similar yet exhibit some differences, demonstrating the effectiveness of the dynamic graph designed in this study.

5. Conclusions

This article presents the CTDGCN model using multi-sensor data to perform RUL prediction. The model does not rely on prior knowledge but does construct a dynamic graph through the fusion of spatio-temporal feature embedding and multisensor measurement signals among sensors. In the CTST encoder, the model is able to extract spatio-temporal features

and build a cross-time sensor dependency network, thereby avoiding the omission of data features at different moments. By utilizing the attenuation mechanism and cross-time pooling layers, the model enhances the extraction of correlations. At last, the encoder's output is passed through a feedback structure for multi-layered predictions, resulting in the final prediction outcomes. In the studies of two cases, the proposed method, which combines dynamic graphs with cross-temporal spatio-temporal information extraction, showed outstanding performance. Given the importance of the sliding window length of time series and the length of cross-temporal periods, their effects on RUL prediction performance are analyzed separately. Ablation experiments are then performed to analyze the contribution of the CTDGCN parts to the prediction performance. Additionally, the role of the adaptive decay adjustment matrix in adjusting across time periods is explored. Finally, the performance variations of the dynamic graph model under different embedding lengths are compared, and two dynamic time periods are also visualized to illustrate the model's behavior. Since this framework does not require prior knowledge to construct the graph, it possesses strong versatility. Therefore, in theory, it can be generalized to various multi-variable time series prediction tasks across different fields, addressing issues such as difficulties in modeling and insufficient extraction of temporal features, which can lead to low prediction accuracy or even difficulty in making predictions. However, despite its superior performance, CTDGCN still has room for improvement. For example, constructing the cross-temporal sensor network introduces some computational complexity, and simplifying the relationships among sensors at different time periods will further enhance the model's versatility. Future work will focus on exploring more straightforward methods to construct the correlations among sensors across time periods.

Acknowledgment

This work was supported by National Key Research and Development Program of China under Grant 2023YFB4704000, National Natural Science Foundation of China under Grant 62073091, Guangdong Basic and Applied Basic Research Foundation under Grant 2023B1515120097, Guangdong Province Science and Technology Innovation Strategic Special Funding under Grant 2023S003042, Maoming City Science and Technology Plan Project under Grant 2021002 and 2024012, and the Talent Introduction Project for Guangdong University of Petrochemical Technology under Grant 2020rc32.

References

1. Yang C, Cai B, Wu Q, et al. Digital twin-driven fault diagnosis method for composite faults by combining virtual and real data. Journal of Industrial Information Integration, 2023, 33: 100469, https://doi.org/10.1016/j.jii.2023.100469.
2. Ma B, Yan S, Wang X, et al. Similarity-based failure threshold determination for system residual life prediction. Eksploatacja i Niezawodnosc – Maintenance and Reliability 2020, 22(3): 520-529, https://doi.org/10.17531/ein.2020.3.15.
3. Shao X, Cai B, Gao L, et al. Data-model-linked remaining useful life prediction method with small sample data: A case of subsea valve. Reliability Engineering & System Safety, 2024, 250: 110323, https://doi.org/10.1016/j.ress.2024.110323.
4. Guan Q, Zhang H, Jia L, Two-stage Remaining Useful Life Prediction Based on the Wiener Process With Multi-feature Fusion and Stage Division. Eksploatacja i Niezawodnosc – Maintenance and Reliability 2024: 26(4), http://doi.org/10.17531/ein/189803.
5. Cai B, Fan H, Shao X, et al. Remaining useful life re-prediction methodology based on Wiener process: Subsea Christmas tree system as a case study. Computers & Industrial Engineering, 2021, 151: 106983, https://doi.org/10.1016/j.cie.2020.106983.
6. Lyu Y, Jiang Y, Zhang Q, et al. Remaining useful life prediction with insufficient degradation data based on deep learning approach. Eksploatacja i Niezawodnosc – Maintenance and Reliability 2021, 23(4): 745-756, http://doi.org/10.17531/ein.2021.4.17.
7. Liu X, Cai B, Yuan X, et al. A hybrid multi-stage methodology for remaining useful life prediction of control system: Subsea Christmas tree as a case study. Expert Systems with Applications, 2023, 215: 119335, https://doi.org/10.1016/j.eswa.2022.119335.
8. Zhang C, Lim P, Qin A. K, et al. Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. IEEE transactions on neural networks and learning systems, 2016, 28(10): 2306-2318, https://doi.org/10.1109/TNNLS.2016.2582798.
9. KARABACAK Y. Deep learning-based CNC milling tool wear stage estimation with multi-signal analysis. Eksploatacja i Niezawodnosc – Maintenance and Reliability 2023, 25(3), https://doi.org/10.17531/ein/168082.

10. Lyu Y, Zhang Q, Chen A, et al. Interval prediction of remaining useful life based on convolutional auto-encode and lower upper bound estimation. Eksploatacja i Niezawodnosc – Maintenance and Reliability 2023, 25(2), https://doi.org/10.17531/ein/165811.
11. Sateesh Babu G, Zhao P, Li X. L. Deep convolutional neural network based regression approach for estimation of remaining useful life. Database Systems for Advanced Applications: 21st International Conference, DASFAA 2016, Dallas, TX, USA, April 16-19, 2016, Proceedings, Part I-21. Springer International Publishing, 2016: 214-228, https://doi.org/10.1007/978-3-319-32025-0_14.
12. Miao H, Li B, Sun C, et al. Joint learning of degradation assessment and RUL prediction for aeroengines via dual-task deep LSTM networks. IEEE Transactions on Industrial Informatics, 2019, 15(9): 5023-5032, https://doi.org/10.1109/TII.2019.2900295.
13. Liu H, Liu Z, Jia W, et al. A novel deep learning-based encoder-decoder model for remaining useful life prediction. International Joint Conference on Neural Networks (IJCNN). IEEE, 2019: 1-8, https://doi.org/10.1109/IJCNN.2019.8852129.
14. Shao X, Cai B, Zou Z, et al. Artificial intelligence enhanced fault prediction with industrial incomplete information. Mechanical Systems and Signal Processing, 2025, 224: 112063, https://doi.org/10.1016/j.ymssp.2024.112063.
15. Liu J, Lei F, Pan C, et al. Prediction of remaining useful life of multi-stage aero-engine based on clustering and LSTM fusion. Reliability Engineering & System Safety, 2021, 214: 107807, https://doi.org/10.1016/j.ress.2021.107807.
16. Zhang J, Jiang Y, Wu S, et al. Prediction of remaining useful life based on bidirectional gated recurrent unit with temporal self-attention mechanism. Reliability Engineering & System Safety, 2022, 221: 108297, https://doi.org/10.1016/j.ress.2021.108297.
17. Wang X, Wang W, Cai X. Automatic measurement of fetal head circumference using a novel GCN-assisted deep convolutional network. Computers in Biology and Medicine, 2022, 145: 105515, https://doi.org/10.1016/j.compbiomed.2022.105515.
18. Shin Y, Yoon Y. PGCN: Progressive graph convolutional networks for spatial–temporal traffic forecasting. IEEE Transactions on Intelligent Transportation Systems, 2024, https://doi.org/10.1109/TITS.2024.3349565.
19. Liang P, Li Y, Wang B, et al. Remaining useful life prediction via a deep adaptive transformer framework enhanced by graph attention network. International Journal of Fatigue, 2023, 174: 107722, https://doi.org/10.1016/j.ijfatigue.2023.107722.
20. Basak S, Sengupta S, Wen S. J, et al. Spatio-temporal AI inference engine for estimating hard disk reliability. Pervasive and Mobile Computing, 2021, 70: 101283, https://doi.org/10.1016/j.pmcj.2020.101283.
21. Li T, Zhao Z, Sun C, et al. Hierarchical attention graph convolutional network to fuse multi-sensor signals for remaining useful life prediction. Reliability Engineering & System Safety, 2021, 215: 107878, https://doi.org/10.1016/j.ress.2021.107878.
22. Wang L, Cao H, Xu H, et al. A gated graph convolutional network with multi-sensor signals for remaining useful life prediction. Knowledge-Based Systems, 2022, 252: 109340, https://doi.org/10.1016/j.knosys.2022.109340.
23. Wei Y, Wu D, Terpenny J. Remaining useful life prediction using graph convolutional attention networks with temporal convolution-aware nested residual connections. Reliability Engineering & System Safety, 2024, 242: 109776, https://doi.org/10.1016/j.ress.2023.109776.
24. Li T, Zhou Z, Li S, et al. The emerging graph neural networks for intelligent fault diagnostics and prognostics: A guideline and a benchmark study. Mechanical Systems and Signal Processing, 2022, 168: 108653, https://doi.org/10.1016/j.ymssp.2021.108653.
25. Ma M, Wang Z, Zhong Z. Transformer Encoder Enhanced by an Adaptive Graph Convolutional Neural Network for Prediction of Aero-Engines' Remaining Useful Life. Aerospace, 2024, 11(4): 289, https://doi.org/10.3390/aerospace11040289.
26. Zhang Y, Li Y, Wei X, et al. Adaptive spatio-temporal graph convolutional neural network for remaining useful life estimation. International joint conference on neural networks (IJCNN). IEEE, 2020: 1-7, https://doi.org/10.1109/IJCNN48605.2020.9206739.
27. Kipf T. N, Welling M. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016, https://doi.org/10.48550/arXiv.1609.02907.
28. Vaswani A, Shazeer N, Parmar N, et al. Attention Is All You Need. arXiv, 2017. https://doi.org/10.48550/arXiv.1706.03762.
29. Weng W, Fan J, Wu H, et al. A Decomposition Dynamic graph convolutional recurrent network for traffic forecasting. Pattern Recognition, 2023, 142: 109670, https://doi.org/10.1016/j.patcog.2023.109670.
30. Yu B, Yin H, Zhu Z. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. arXiv preprint arXiv:1709.04875, 2017, https://doi.org/10.24963/ijcai.2018/505.
31. Wang Y, Xu Y, Yang J, et al. Fully-Connected Spatial-Temporal Graph for Multivariate Time-Series Data. Proceedings of the AAAI Conference on Artificial Intelligence. 2024, 38(14): 15715-15724, https://doi.org/10.1609/aaai.v38i14.29500.
32. Saxena A, Goebel K, Simon D, et al. Damage propagation modeling for aircraft engine run-to-failure simulation. International conference

on prognostics and health management. IEEE, 2008: 1-9, https://doi.org/10.1109/PHM.2008.4711414.

33. The Prognostics and Health Management Society (PHM Society), https://www.phmsociety.org/competition/phm/10, 2010.1.
34. Li J, Li X, He D. A directed acyclic graph network combined with CNN and LSTM for remaining useful life prediction. IEEE Access, 2019, 7: 75464-75475, https://doi.org/10.1109/ACCESS.2019.2919566.
35. Jiang Y, Lyu Y, Wang Y, et al. Fusion network combined with bidirectional LSTM network and multiscale CNN for remaining useful life estimation. 2020 12th International Conference on Advanced Computational Intelligence (ICACI). IEEE, 2020: 620-627, https://doi.org/10.1109/ICACI49185.2020.9177774.
36. Li X, Jiang H, Liu Y, et al. An integrated deep multiscale feature fusion network for aeroengine remaining useful life prediction with multisensor data. Knowledge-based systems, 2022, 235: 107652, https://doi.org/10.1016/j.knosys.2021.107652.
37. Chen W, Liu C, Chen Q, et al. Multi-scale memory-enhanced method for predicting the remaining useful life of aircraft engines. Neural Computing and Applications, 2023, 35(3): 2225-2241, https://doi.org/10.1007/s00521-022-07378-z.
38. Wang T, Li X, Wang W, et al. A spatiotemporal feature learning-based RUL estimation method for predictive maintenance. Measurement, 2023, 214: 112824, https://doi.org/10.1016/j.measurement.2023.112824.
39. Zhang J, Li X, Tian J, et al. An integrated multi-head dual sparse self-attention network for remaining useful life prediction. Reliability Engineering & System Safety, 2023, 233: 109096, https://doi.org/10.1016/j.ress.2023.109096.
40. Zhang Y, Su C, Wu J, et al. Trend-augmented and temporal-featured Transformer network with multi-sensor signals for remaining useful life prediction. Reliability Engineering & System Safety, 2024, 241: 109662, https://doi.org/10.1016/j.ress.2023.109662.
41. Guo J, Lei S, Du B. MHT: A multiscale hourglass-transformer for remaining useful life prediction of aircraft engine. Engineering Applications of Artificial Intelligence, 2024, 128: 107519, https://doi.org/10.1016/j.engappai.2023.107519.
42. Huang C. G, Huang H. Z, Li Y. F. A bidirectional LSTM prognostics method under multiple operational conditions. IEEE Transactions on Industrial Electronics, 2019, 66(11): 8792-8802, https://doi.org/10.1109/TIE.2019.2891463.
43. Wu Z, Yu S, Zhu X, et al. A weighted deep domain adaptation method for industrial fault prognostics according to prior distribution of complex working conditions. IEEE Access, 2019, 7: 139802-139814, https://doi.org/10.1109/ACCESS.2019.2943076.
44. Xu X, Wu Q, Li X, et al. Dilated convolution neural network for remaining useful life prediction. Journal of Computing and Information Science in Engineering, 2020, 20(2): 021004, https://doi.org/10.1115/1.4045293.