Position Encoding Based Convolutional Neural Networks for Machine Remaining Useful Life Prediction

Ruibing Jin, Min Wu, *Senior Member, IEEE*, Keyu Wu, Kaizhou Gao, *Member, IEEE*, Zhenghua Chen, *Senior Member, IEEE*, and Xiaoli Li, *Senior Member, IEEE*

***Abstract*—Accurate remaining useful life (RUL) prediction is important in industrial systems. It prevents machines from working under failure conditions, and ensures that the industrial system works reliably and efficiently. Recently, many deep learning based methods have been proposed to predict RUL. Among these methods, recurrent neural network (RNN) based approaches show a strong capability of capturing sequential information. This allows RNN based methods to perform better than convolutional neural network (CNN) based approaches on the RUL prediction task. In this paper, we question this common paradigm and argue that existing CNN based approaches are not designed according to the classic principles of CNN, which reduces their performances. Additionally, the capacity of capturing sequential information is highly affected by the receptive field of CNN, which is neglected by existing CNN based methods. To solve these problems, we propose a series of new CNNs, which show competitive results to RNN based methods. Compared with RNN, CNN processes the input signals in parallel so that the temporal sequence is not easily determined. To alleviate this issue, a position encoding scheme is developed to enhance the sequential information encoded by a CNN. Hence, our proposed position encoding based CNN called PE-Net is further improved and even performs better than RNN based methods. Extensive experiments are conducted on the C-MAPSS dataset, where our PE-Net shows state-of-the-art performance.**

***Index Terms*—Convolutional neural network (CNN), deep learning, position encoding, remaining useful life prediction.**

I. Introduction

PROGNOSTIC health management (PHM) plays an important role in industry, and improves the stability and reliability of industry equipments. In PHM, remaining useful life (RUL) prediction is often used to prevent industrial systems from unnecessary downtime, and thus, reduces maintenance costs [1]. Recently, neural networks have been widely studied, and they have shown impressive performance in different applications, such as computer vision [2], [3], nature language processing [4], [5] and optimization [6]–[8]. Without the expert knowledge of a specific field, neural networks are able to learn an effective model given enough training data. For example, in industry, the control system may be complex, which often poses a challenge to formulate a model for optimization. To solve this problem, some neural networks [6]–[8] are often proposed to simplify the complex control systems and achieve better performances. With the remarkable progress of deep learning, the existing neural network becomes more deeper. Many deep learning based approaches which consist of these neural networks, have been proposed [1], [9]–[22] for the RUL prediction, and achieve notable performances on this task. Currently, existing deep learning based methods fall into three categories, convolutional neural network (CNN) based methods, recurrent neural network (RNN) based methods, and the hybrid of CNN and RNN based methods.

In CNN based methods [10], [13], [14], several convolutional layers and pooling layers are stacked together as a deep neural network, which directly extracts feature representation from raw time series signals. Then, the fully connected layers are used as a regressor to predict the final results. Benefiting from this deep neural network architecture, CNN based methods [10], [13], [14], [23] are able to provide much useful information for RUL prediction. With the features automatically learned from a CNN, the regressor can then accurately predict the machine RUL. Time series signals are usually the inputs for the RUL prediction. It is thus important to capture the sequential information in time series signals for predicting the RUL. Since CNN is originally designed to process data in a parallel way, several studies [1], [11], [15]–[17], [21] argue that CNN may not be suitable to exploit the sequential information compared with RNN.

With this viewpoint, some RNN based approaches [1], [11], [15]–[17], [21], [22], [24], [25] are proposed. Usually, RNN receives the signal in a recurrent way, which enables it to capture sequential information, providing more accurate prediction results. However, limited by the recurrent forward

Manuscript received April 26, 2022; revised May 24, 2022; accepted May 26, 2022. This work was supported by National Research Foundation of Singapore, AME Young Individual Research Grant (A2084c0167). Recommended by Associate Editor Shuai Li. *(Corresponding author: Zhenghua Chen.)*

Citation: R. B. Jin, M. Wu, K. Y. Wu, K. Z. Gao, Z. H. Chen, and X. L. Li, "Position encoding based convolutional neural networks for machine remaining useful life prediction," *IEEE/CAA J. Autom. Sinica*, vol. 9, no. 8, pp. 1427–1439, Aug. 2022.

R. B. Jin, M. Wu, K. Y. Wu, Z. H. Chen, and X. L. Li are with the Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore 138632, Singapore (e-mail: jin_ruibing@i2r.a-star.edu.sg; wumin@i2r.a-star.edu.sg; wu_keyu@i2r.a-star.edu.sg; chen0832@e.ntu.edu.sg; xlli@i2r.a-star.edu.sg).

K. Z. Gao is with the School of Computer Science, Liaocheng University, Liaocheng 252000, and also with the Macau Institute of System Engineering, Macau University of Science and Technology, Taipa, Macao 999078, China (e-mail: gaokaizh@aliyun.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2022.105746

scheme in RNN, RNN based methods cannot be trained in parallel. The training process of deep RNN becomes time-consuming. In practice, most RNN based methods only stack two or three RNN layers. Since shallow networks may not be able to generate strong feature representations, the performances of RNN based methods are thus hindered. To address this issue, some hybrid of CNN and RNN based methods [12], [18], [19] are proposed, where a deep CNN is used as a feature extractor, and a shallow RNN is applied to exploit the sequential information. In these hybrid approaches, although a deep CNN is adopted for feature generation, sequential information is still captured by a shallow RNN. However, sequential information may not be fully exploited, which causes these hybrid based methods to be sub-optimal.

To alleviate this problem, we propose a new CNN based method called the **p**osition **e**ncoding based **net**work (PE-Net) in this paper. Firstly, we rethink the common viewpoint held in RUL prediction, and investigate how to improve the CNN's capacity of capturing sequential information. Through a series of analysis, we find that existing CNN based approaches [10], [13], [14], [23] are limited by three factors. 1) Existing CNN based methods neglect the relationship between the receptive field and the capability of capturing sequential information. Actually, sequential information captured by CNN can be improved via enlarging the receptive field for the last convolutional layer in a CNN. 2) The architecture of existing CNN based methods do not follow the classic principle which is widely adopted by famous CNNs like VGG16 [2], GoogleNet [26], ResNet [3] and MobileNet [27]. According to this principle, the kernel size should be smaller and the channel number should be larger, when the CNN depth increases. Extensive experiments have demonstrated that this principle can enhance CNN's capability of feature learning. However, existing CNN based RUL prediction methods are not designed according to this principle [10], [13], [14], [23]. They simply stack the same convolutional layer repeatedly, which increases the difficulty in training and degenerates the quality of the feature map produced by CNN. 3) Feature normalization is not considered by existing CNN based methods. Without feature normalization, overfitting easily happens during the CNN's training process. This further limits the performances of CNN based methods.

Motivated by these observations, we propose a series of novel CNNs with carefully-designed network architectures. Our new CNNs are proposed in the following three aspects. Firstly, we enlarge the CNN's receptive field for the final convolutional layer. This enables our CNN to capture more sequential information compared with existing CNN based methods. Additionally, our proposed CNNs are designed according to the classic principles, which improves the quality of feature map produced by our CNN. Furthermore, our proposed CNN is a feature normalization based network, which adopts the batch normalization technique to improve the RUL prediction accuracy. To the best of our knowledge, we are the *first* to demonstrate the importance of the above aspects to design CNNs for the RUL prediction. With these modifications, our proposed CNNs show satisfactory performances for the RUL prediction, and are able to perform comparably to RNN based methods.

Apart from it, to enhance the encoded sequential information by the CNN, we propose a position encoding scheme according to [4]. The original position encoding scheme [4] is proposed for nature language processing, where a pre-defined position vector is directly added to a feature vector produced by an off-the-shelf word embedding. However, there is no off-the-shelf word embedding in the RUL field. To adopt the position encoding scheme into the RUL task, we firstly propose two transformations which transform both raw input signals and the pre-defined position vectors into the same latent space. Then, we propose two different fusion methods to effectively combine the transformed position information with the transformed input signals together. With our proposed position encoding scheme, our PE-Net can fully exploit the sequential information in time series signals. Through experiments, our proposed PE-Net surpasses existing methods and sets new state-of-the-art performances for the RUL prediction on the C-MAPSS dataset.

Overall, our contributions can be summarized as follows:

1) We first investigate various CNN architectures for capturing sequential information. According to our analysis, a series of novel CNNs are proposed for RUL prediction.

2) To further enhance the capability of capturing sequential information, we propose a position encoding based CNN called PE-Net for more accurate RUL prediction.

3) Extensive experiments are conducted, and our PE-Net shows state-of-the-art performances on the C-MAPSS dataset.

The rest of the paper is organized as follows: Section II discusses related work. Our PE-Net method is presented in Section III. Experimental results on public datasets and related discussions are provided in Section IV. We present conclusions and future work in Section V.

II. Literature Review

Methods in RUL prediction can be roughly divided into two categories: model based [28], [29] and data driven methods [30]–[32].

Model Based Methods: Model based methods are proposed according to the physical properties of the industrial systems [28], [29]. These approaches investigate the relationship between the sensor data and the target output, explicitly. However, these model based methods become unpractical, when the industrial systems are complex.

Data Driven Methods: Compared with model based methods, data driven methods do not require physical knowledge on the industrial systems. They can be easily applied to industrial systems by collecting enough sensor data. For data driven methods, they can be further divided into two classes: conventional machine learning based methods [33] and deep learning based methods. Since our work is proposed based on the deep learning technology, we focus on discussing deep learning based methods in this paper.

In deep learning based methods, a CNN based method [14] is firstly proposed to explore how to apply CNN for the RUL prediction. In this method, the proposed CNN consists of two 2D convolutional layers, two pooling layers and one fully connected layer. To adapt the 2D convolutional layer with the

time series signal, the original signal is segmented by a sliding window strategy into some two-dimensional data matrix, where each row indicates the time series signal from one sensor, and each column represents the time step. Instead of using 2D convolutional layers, Li *et al*. [23] utilize 1D convolutional layers to extract features from the original signals. A deeper CNN is proposed in [23], where five 1D convolutional layers are used to produce the feature representation and one fully connected layer is used to predict the final results. In the paper [13], a double-convolutional neural network (D-CNN) is proposed. D-CNN is composed of two convolutional neural networks. It divides the RUL prediction into two stages: fault point detection and RUL prediction. One CNN firstly predicts the fault point, and another CNN is used to produce the RUL according to the fault point estimation.

Although CNN based methods achieve remarkable results, their performances are not satisfactory. CNN processes input signals in a parallel way. Compared with CNN, RNN receives the input signal recursively, which demonstrates that RNN may be more suitable to process time series signals. Deep long short-term memory (LSTM) [15] proposes a network consisting of two LSTM layers and two fully connected layers. By using LSTM, Deep LSTM [15] shows better performances than CNN based methods. LSTM only receives information in a forward manner. To further improve the captured sequential information, [16] proposes a bi-directional LSTM based approach. The bi-directional LSTM (Bi-LSTM) is able to receive time series signals in both a forward and backward manner, which improves the exploited sequential information, enhancing RUL prediction accuracy. Following the method in [16], Huang *et al*. [17] combines operational conditions with original input signals together in a bi-directional LSTM network. The method [17] effectively uses operational conditions to improve the performance on the RUL prediction. ATS2S [24] proposes a self-supervised method based on a sequence-to-sequence approach to improve performance. Chen *et al*. [1] propose an attention-based LSTM method for the RUL prediction, where handcrafted features are integrated with learned features by LSTM. Based on the fused features, RUL prediction accuracy is improved. In the [11], an attention-based gated recurrent unit (GRU) network is proposed for RUL prediction.

RNN based methods (including LSTM and GRU) achieve better performance than CNN based methods. However, since RNN receives input signals recursively, it is impractical to train a very deep RNN network. To utilize deep network features, some hybrid of CNN and LSTM based methods [12], [18], [19] are proposed. Although these approaches show impressive performance, their RNN parts are shallow modules, which limit their performances. To solve this problem, we investigate a series of effective CNNs for the RUL prediction in this paper. With novel CNN architectures, our CNNs can be easily trained and capture the sequential information effectively. Moreover, based on [4], we develop a position encoding scheme in our CNNs to better model the sequential information and thus achieving better performances.

III. Method

In this section, we first investigate the relationship between the CNN architecture and its capability of modelling sequential information. Based on our analysis, a series of new convolutional neural networks are proposed. Furthermore, a position encoding scheme is developed to further enhance the encoded sequential information.

Following the protocol of existing methods [10], [14], a sliding window technology is applied to original time series signals. These signals are segmented into an input matrix denoted as $X \in \mathbb{R}^{T \times C}$, where T is the segmented temporal length and C represents the number of sensors. Given segmented input X, a designed neural network N is expected to produce the final output y_p. This can be formulated as

$$y_p = N(X). \tag{1}$$

In the following parts, we discuss how to design a CNN which can effectively capture sequential information, and provide an accurate estimation for RUL.

A. Network Architecture Analysis

The performance of a CNN is affected by many factors. In this section, we discuss the CNN's architecture with respect to convolutional kernel selection, layer type, receptive field and network depth.

1) Convolutional Kernel Selection: There are three conventional convolutions in CNN: 1D convolution, 2D convolution and 3D convolution. 3D convolution is often used in video related tasks, such as action recognition [34] and video object detection [35], [36]. Since time series signals in the RUL prediction are segmented into 2D matrix, 3D convolution is apparently not suitable for the RUL prediction.

For RUL prediction, existing CNN based methods can be divided into two categories in view of convolutional layer type: 1D convolution based methods [10], [13], [23] and 2D convolution based methods [12], [14]. In this part, we perform a comparison between 1D convolution and 2D convolution according to the characteristic of time series signals, and investigate which convolution operation is more suitable for the RUL prediction.

For the 1D convolution operation, the convolution kernel is moved along one direction. Given an input matrix $\mathbf{M}_{in} \in \mathbb{R}^{L_{in} \times C_{in}}$, the corresponding output $\mathbf{M}_o \in \mathbb{R}^{L_o}$ is computed as

$$\mathbf{M}_o(t) = b + \sum_{\tau=-(K-1)/2}^{\tau=(K-1)/2} \mathbf{W}(\tau) \cdot \mathbf{M}_{in}(t+\tau) \tag{2}$$

where K is kernel size, b denotes a learnable bias and $\mathbf{W}$ is a learnable kernel weight matrix in the convolutional layer.

In Fig. 1, the 1D convolution operation is applied on time series signals, where each sensor signal is indicated by a unique color. The vertical axis indicates the sensor index, and the horizontal axis represents the time step.

As shown at the left side of Fig. 1, the kernel is moved along the time axis in the single sensor case. When the number of sensors increases, different types of sensors are put along the channel axis. According to the convolution defini-

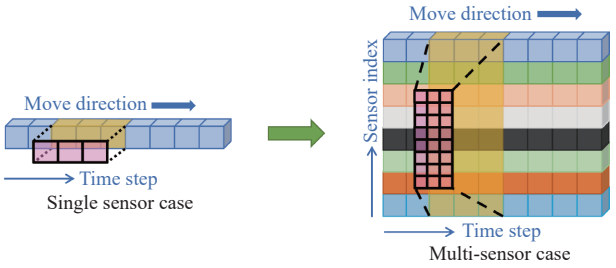


Fig. 1. 1D convolution operation applied on time series. In the single sensor case, as shown at the left side, a convolution kernel is moved along the time axis. When the number of sensors increases, according to the convolution definition, the number of channels of the kernel is extended to the same as that of input, which is illustrated at the right side.

tion, the number of channels of the kernel should be the same as that of the input. As displayed at the right side of Fig. 1, we extend the kernel along the channel axis and still move the kernel along the time axis.

In comparison, 2D convolution operation moves the convolution kernel in a two-dimension space. Given a two-dimension matrix $\mathbf{M}_{in}$, 2D convolution operation computes the output $\mathbf{M}_o^{'} \in \mathbb{R}^{L_o \times C_o}$ according to

$$\mathbf{M}_o^{'}(t,s) = b + \sum_{\tau=-\frac{K-1}{2}}^{\tau=\frac{K-1}{2}} \sum_{\sigma=-\frac{K-1}{2}}^{\sigma=\frac{K-1}{2}} \mathbf{W}(\tau,\sigma) \cdot \mathbf{M}_{in}(t+\tau, s+\sigma). \quad (3)$$

We visualize the 2D convolution operation applied to time series signals in Fig. 2. According to the left side of Fig. 2, in the single sensor case, since there is only one type of sensor, the 2D convolution is actually equal to 1D convolution. In the multi-sensor case, different from 1D convolution, 2D convolution kernel is moved along two directions.

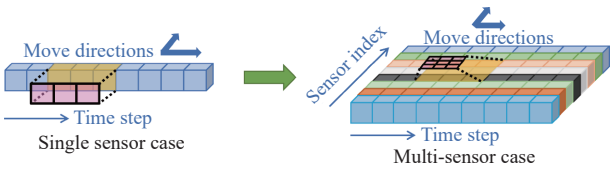


Fig. 2. 2D convolution operation applied on time series. Different from the 1D convolution operation, the 2D convolution operation moves the convolution kernel on two directions: the time step direction and the sensor index direction. In the single signal case, since only one type of sensor exists, as displayed at the left side, the 2D convolution is equal to 1D convolution. In the multi-sensor case, as illustrated at the right side, the 2D convolution moves the kernel on a space.

To investigate the suitable convolution operation for time series signals, we need to analyze the characteristics of time series signals. Some example time series signals are shown in Fig. 3. As shown in Fig. 3 , signals from different sensors show significantly diverse properties. The convolutional layer shares parameters and aims to learn the translation invariant features. When 2D convolution is applied to time series signals, different sensor signals are formulated into a two-dimensional matrix. It is difficult to exploit the invariant information due to diverse properties existing in different sensor signals. Compared with 2D convolution, 1D convolution operations move the convolution kernel along the time axis, instead of both the sensor axis and time axis. It is much easier for 1D convolution kernels to learn generic features from time series signals.

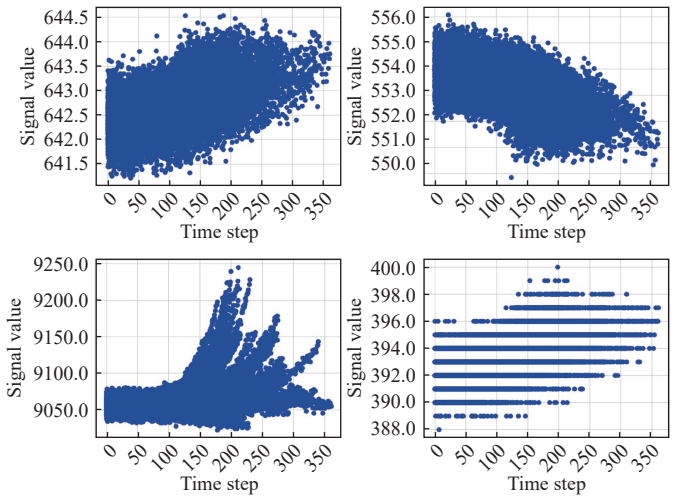


Fig. 3. Visualization for different time series signals. It can be found that different signals show different properties.

2) Receptive Field for Modeling Sequential Information: Although RNN's recursive forward scheme enables it to capture sequential information, it is unpractical to establish a deep RNN. This limits the feature representation for the RUL prediction. Compared with RNN, a deep CNN is much easier to train. In this part, we investigate how to design a deep CNN which can effectively encode sequential information from time series signals.

Since the kernel size is often much smaller than the temporal length of the times series input, people believe that CNN cannot fully exploit sequential information. However, the CNN's capability of modelling the sequential information is not decided by the kernel size. This capability is actually affected by the receptive field of CNN's final feature map.

As show in Fig. 4, the feature maps from three consecutive convolutional layers are illustrated, where the kernel size in the corresponding CNN is set as three. For convenience, we use r_n to represent the receptive field at the nth feature map

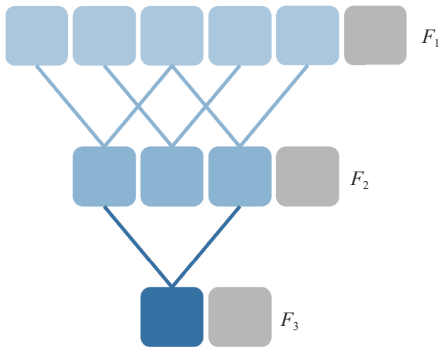


Fig. 4. Visualization for the receptive field in a CNN. It shows that the receptive field can be enlarged by stacking several convolutional layers.

F_n. In Fig. 4, F_1 indicates the raw input signal, and r_1 is 1. The second feature map F_2 is generated by a convolutional layer with a kernel (3×1), and the r_2 is 3. At the final feature map F_3, the receptive field r_3 becomes 5, though the convolutional kernel size is 3. This demonstrates that each feature vector at F_3 is able to capture information across 5 time steps in the raw input signal. Furthermore, we can stack more convolutional layers to encode more temporal information. This indicates that actually, CNN is able to capture long dependency or sequential information by stacking more convolutional layers.

In a CNN, we can compute the receptive field of the final produced feature map, r_f, recursively, according to

$$r_f = \sum_{l=1}^{L}((k_l - 1)\prod_{i=1}^{l-1} s_i) + 1 \tag{4}$$

where s_i represents the stride value at the ith layer and k_l denotes the kernel size at the lth layer.

To fully encode the sequential information, we propose to design a CNN whose r_f is greater than or equal to the temporal length T of the input signal X.

3) Network Depth and Kernel Size: To ensure the CNN's sequential information capacity, a large receptive field is required. To fulfill this requirement, we have two solutions: a large kernel size with a shallow network, and a small kernel size with a deep network. In VGG-Net [2], more non-linear layers make the decision function of a CNN more discriminative. According to ZF-Net [37], the feature quality is improved when the kernel size and stride become small. Based on these observations, we design a new CNN with small kernel size and increased depth.

B. Proposed Convolutional Neural Networks

According to the discussions above, we propose a series of deep CNNs for the RUL prediction in this part.

Various famous CNNs, such as VGG16 [2], GoogleNet [26], ResNet [3] and MobileNet [27], follow a principle to design their network architecture, i.e., the kernel size should be smaller and the channel number should be larger when the CNN depth increases. However, existing CNN based methods for RUL prediction [10], [13], [14], [23] do not follow this classic principle, which limits their performance. In comparison, our proposed CNNs can produce a high-qualification feature map, by following this design principle.

Additionally, existing CNN based methods [10], [13], [14], [23] only design a shallow CNN for RUL prediction. Considering that the internal covariate shift becomes serious in a deep CNN [38], we adopt a batch normalization (BN) layer [38] in our proposed CNN. The BN layer can be formulated as

$$y = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} * \gamma + \beta \tag{5}$$

where $E[x]$ and $Var[x]$ are the mean and standard-deviation for the input, respectively. γ and β indicate two learnable parameters. In the training process, the BN layer computes the mean and variance according to input signals. By using the BN layer, our proposed CNN can be more stable, and thus is able to learn better feature representations.

Furthermore, to improve the non-linearity of our proposed CNN, we choose the rectified linear unit (ReLU) as the activation function, which is defined as

$$f(x) = x^+ = \max(0, x). \tag{6}$$

In our proposed CNN, each convolutional layer is followed by a BN layer and a ReLU layer, which is illustrated in Fig. 5. We regard this three layer combination as a convolution unit and use it to design our CNNs.

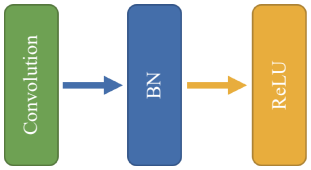


Fig. 5. The convolution unit in our proposed CNN. This unit consists of a convolutional layer, a batch normalization layer and a rectified linear layer.

In Table I, a series of CNNs are proposed for the RUL prediction. Convolutional parameters are indicated as $(k_1 \times k_2, C, s)$, where $k_1 \times k_2$ is the kernel size, C denotes the channel number and s represents the stride value. Each convolutional layer is followed by a BN layer and a ReLU layer. Instead of using the pooling layer, we down sample a feature map by setting the stride value as 2 in some convolutional layers. We also list the corresponding receptive field r_f for the produced feature map by the final convolutional layer. After that, two fully connected (denoted as FC) layers are applied as a decoder, where the hidden number for each layer is 256. Finally, we use an FC layer to predict the final results.

TABLE I
OUR PROPOSED CNN FOR THE RUL PREDICTION. THE PARAMETER FOR A CONVOLUTIONAL LAYER IS REPRESENTED AS (KERNEL SIZE, CHANNEL NUMBER, STRIDE VALUE)

Layer name	CNN-A	CNN-B	CNN-C	CNN-D
conv1	5×1, 16, 2	5×1, 16, 2	5×1, 16, 2	5× 1, 16, 2
conv2-x	3×1, 64, 1	3×1, 64, 1	3×1, 64, 1 3×1, 64, 1	3×1, 64, 1 3×1, 64, 1
conv3-x	N.A.	3×1, 128, 2	3×1, 128, 2 3×1, 128, 1	3×1, 128, 2 3×1, 128, 1
conv4-x	N.A.	3×1, 256, 2	3×1, 256, 2	3×1, 256, 2 3×1, 256, 1
r_f	9	21	33	49
FC	256-d × 2			
regressor	1-d			

As listed in Table I, we propose four different CNNs. In Table I, it can be seen that our proposed CNNs are designed following the principle discussed above, where we only set the kernel size as 5 at the first layer, and the kernel sizes for other layers are set as 3. For the channel number, we gradually increase it from 16 to 256. CNN-A consists of two convolutional units. We set the stride value for the conv1 layer as 2 for down sampling feature map. In CNN-B, two more

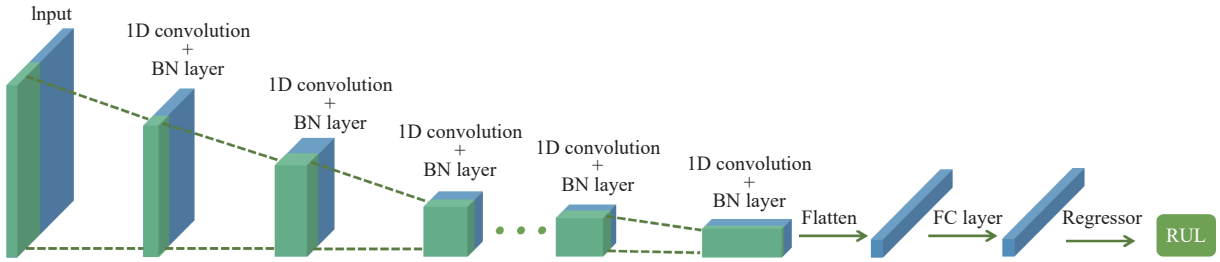


Fig. 6. The work flow of our proposed CNNs. With a careful design, in our proposed CNN, each convolutional layer is followed by a batch normalization layer, and the variation of the produced feature map follows the classic principle: the channel number increases, and the height and width decrease with the CNN depth increasing.

convolutional units are added. The stride values for these two additional convolutional layers are set as 2. The receptive field for CNN-B is increased by stacking more layers. To further improve the receptive field of CNN, we stack more layers, and propose CNN-C and CNN-D. To present our proposed CNN clearly, the work flow is visualized in Fig. 6. As shown in Fig. 6, the channel number of the feature map gradually becomes deep, and the height and width gradually decrease, since the feature map is down-sampled and the channel number increases when the CNN depth increases. In our CNN, the BN layer is integrated with 1D concolutional layer, which solves the issue of internal covariate shift in CNN. In our experiment part, we use these four CNNs with different configurations to verify our hypothesis above.

C. Position Encoding Scheme

In our proposed CNNs, convolutional layers with large receptive fields can capture sequential information. However, the receptive fields for the first several layers are still limited. To further improve the encoded sequential information, we propose a position encoding scheme according to [4].

Intuitively, we can use the position index as the encoded position information $\mathcal{P}_s$, which can be computed as

$$\mathcal{P}_s(p) = p \tag{7}$$

where p is the position index. However, the useful information provided by this simple method is limited.

Alternatively, we propose projecting the position information into a high dimensional space. A p_d-dimension position encoding basis $\mathbf{p}_b$ is firstly defined as

$$\mathbf{p}_b(i) = 10000^{i/p_d} \tag{8}$$

where i is the dimension index. Based on $\mathbf{p}_b$, we can easily get a high-dimensional position encoding vector $\mathcal{P}' \in \mathbb{R}^d$ as

$$\mathcal{P}'(p,i) = p/\mathbf{p}_b(i). \tag{9}$$

To further increase the non-linearity of the position encoding function, we apply different operations on different dimension indexes, which can be formulated as

$$\mathcal{P}(p,i) = f(\mathcal{P}'(p,i),i) \tag{10}$$

where $f(*,i)$ applies different operations on different value i. Specifically, in this paper, $f(*,i)$ is defined as

$$f(*,i) = \begin{cases} \sin(*), & i\%2 = 1 \\ \cos(*), & i\%2 = 0 \end{cases} \tag{11}$$

where i represents the dimension index in our proposed position encoding scheme. When we use sine and cosine functions, the $\mathbf{p}_b(i)$ is modified as $\mathbf{p}_b(i) = 10000^{2*[i/2]/p_d}$.

With the above position encoding scheme, there may be a domain gap between our encoded position information and the input signals. To address this issue, we propose a novel position encoding scheme, where two transformations, $\Phi(*)$ and $\Psi(*)$ are proposed to translate the encoded position information and input signals into the same high dimension space, and then these two translated features are fused together. This fusion process can be formulated as

$$\mathcal{F} = \Omega(\Phi(\mathcal{P}), \Psi(X)) \tag{12}$$

where $\mathcal{F}$ indicates the fused feature vector and Ω is the fusion function. By fusing the position information with input signals, our proposed CNN can effectively capture the sequential information in time series signals.

Fig. 7 shows the proposed fusion module to combine both the position encoding vectors and segmented input signals. In particular, two convolutional layers are used as two transformations $\Phi(P)$ and $\Psi(X)$. Therefore, our proposed position encoding scheme is different from the original version [4] in three aspects. 1) There is no off-the-shelf word embedding in the RUL prediction. To address this issue. we propose a transformation $\mathcal{P}$ to adaptively convert the raw time series signals into feature vectors for the position embedding fusion. 2) Instead of directly adding the produced feature vector to the position vector, we propose another transformation $\Psi(X)$ to project the position vector into a latent space. 3) We propose a fusion module which is represented as $\Omega(*)$ in (12) to combine these two transformed feature vectors together. In the fusion module, we propose two types of fusion methods: concatenation and element-wise addition. After the fusion operation, a batch normalization layer is added to improve the generalization of the encoded output.

D. Optimization

We regard the RUL prediction as a regression problem and use the mean square error (MSE) loss to optimize our proposed neural networks. MSE is defined as

$$\text{MSE Loss} = \frac{1}{N}\sum_{i=1}^{N}(r_i - g_i)^2 \tag{13}$$

where N is the number of samples, r_i denotes the neural network prediction, and g_i indicates the true RUL value.

IV. Experiments

In this section, extensive experiments are conducted to

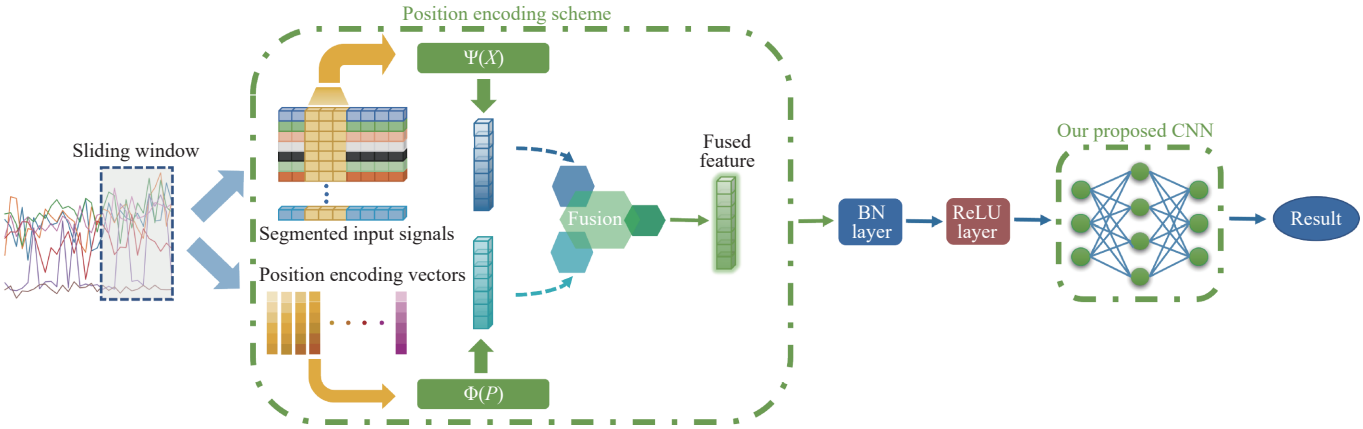


Fig. 7. Fusion module for combining position encoding vectors and segmented input signals. With the sliding window method, the raw signals are firstly segmented as the input signals. Then, the segmented input signals and the position encoding vectors are forwarded to two transformations, respectively. After that, we fuse these two transformed features via our proposed fusion module to produced the fused feature vector. Finally, we forward this fused feature vector into our proposed CNN to predict the RUL result.

verify the performance of the proposed CNNs and the position encoding scheme in our PE-Net.

A. Experimental Dataset and Setting

1) Dataset: The *C-MAPSS* Dataset [39] is widely used in the RUL prediction to evaluate the performance of models. Following the previous studies [1], [14], [17], we also select the *C-MAPSS* Dataset to evaluate our proposed method. It records the signals from 21 sensors installed on aircraft engines, which is illustrated in Fig. 8 . These time series signals describe the degradation process of the aircraft engine. There are four subsets: FD001, FD002, FD003 and FD004 in this dataset, where each subset is divided into a training set and a test set. The details for this dataset are listed in Table II. The maximum time step for each trajectory is different. In FD001, the maximum time step is from 128 to 362. In FD002, the maximum time step is from 126 to 378. The maximum time step varies from 137 to 525 in FD003. In FD004, the maximum time step is from 126 to 554.

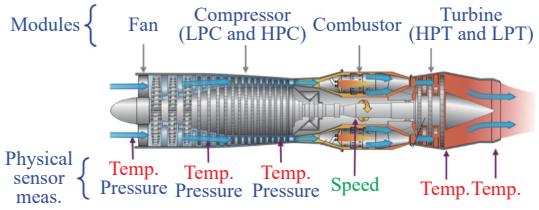


Fig. 8. Illustration for an aircraft engine. Multiple sensors are installed on an engine to measure different signals, such as temperature and speed.

As listed in Table II , the *C-MAPSS* Dataset provides operation condition information and faulty types. There are hundreds of training and testing engine data trajectories. Training trajectories record all the run-to-failure sensor data. Testing trajectories only include a certain period of sensor data during the degradation process. Among four subsets, FD001 is the simplest subset, which only includes one operation condition, one fault mode, 100 training trajectories and 100 testing trajectories. FD004 is the most difficult subset, and involves 6 operation conditions, 2 fault modes, 248 training trajectories and 249 testing trajectories.

TABLE II
DATAILS FOR *C-MAPSS* DATASET

Subsets	Conditions	Fault modes	Training trajectories	Testing trajectories
FD001	1	1	100	100
FD002	6	1	260	259
FD003	1	2	100	100
FD004	6	2	248	249

2) Data Preprocessing: The sensor data provided in the *C-MAPSS* Dataset is redundant. Following methods [1], [14], [17], we remove the data from sensors indexed 1, 5, 6, 10, 16, 18, and 19. The sliding window is widely used for data segmentation [1], [10]. We set the window length as 30 and the step as 1. The piece-wise linear RUL model [1], [10] is also adopted to generate the RUL labels, where the maximal RUL is set as 125. The learning rate is 0.001 and *Adam* is chosen to optimize our proposed neural network.

3) Evaluation Metrics: According to previous studies [1], [14], [17], two metrics, namely RMSE and scoring function [14], are used to evaluate the performance of our method. The RMSE is defined as

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(r_i - g_i)^2} \tag{14}$$

and the scoring function is computed according to

$$S = \begin{cases} \sum_{i=1}^{N}(e^{\frac{g_i - r_i}{13}} - 1), & r_i < g_i \\ \sum_{i=1}^{N}(e^{\frac{r_i - g_i}{10}} - 1), & r_i \geq g_i. \end{cases} \tag{15}$$

B. Comparison With Other Methods

Our experimental results show that our CNN-C and CNN-D achieve similar performances. In this subsection, we compare

TABLE III
COMPARISON WITH OTHER METHODS

Dataset	FD001		FD002		FD003		FD004	
Evaluation	RMSE	Score	RMSE	Score	RMSE	Score	RMSE	Score
Deep LSTM [15]	16.14	338.00	24.49	4450.00	16.18	852.00	28.17	5550
Chen's Hybrid [1]	14.53	322.44	N.A.	N.A.	N.A.	N.A.	27.08	5649.14
Huang's BLSTM [17]	N.A.	N.A.	25.11	4793.00	N.A.	N.A.	26.61	4971
TCMN [19]	23.57	1220.00	20.45	3100.00	21.17	1300.00	21.03	4000.00
GAN [40]	18.54	1467.00	20.06	4085.00	19.28	1488.00	20.88	3872.00
MODBNE [41]	15.04	334.23	25.05	5585.34	12.51	421.91	28.66	6557.62
Babu's CNN [14]	18.45	1286.70	30.29	13570.00	19.82	1596.20	29.16	7886.40
Li's CNN [23]	**12.61**	**273.70**	22.36	10412.00	12.64	284.10	23.31	12466.00
KDnet [10]	13.68	362.08	**14.47**	929.20	12.95	327.27	15.96	1303.19
Our CNN-C	15.69	316.32	16.24	934.11	13.15	275.62	16.29	1204.75
Our PE-Net	13.98	280.87	14.69	**881.73**	**12.33**	**272.85**	**15.40**	**1103.18**

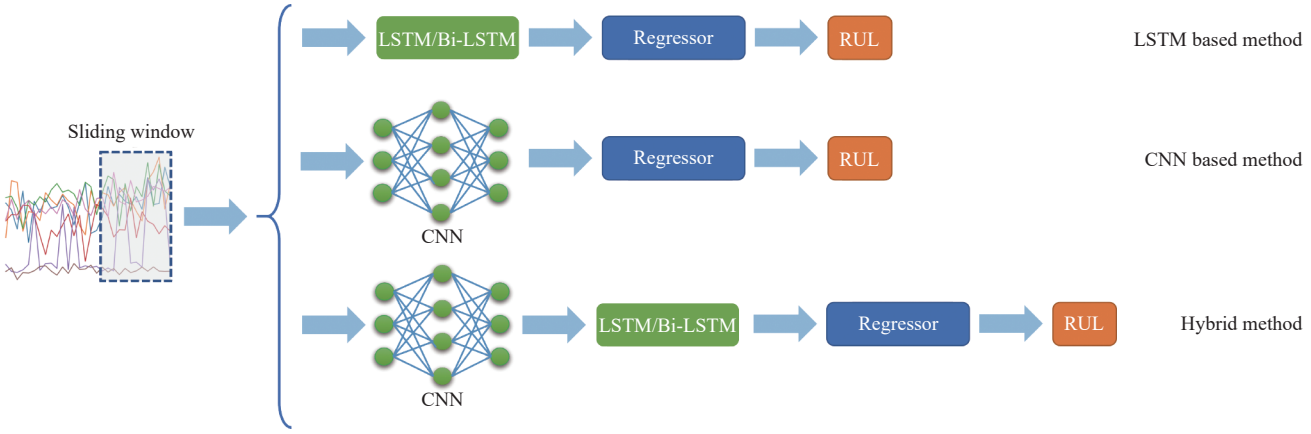


Fig. 9. Illustration for different types of methods. These methods can be roughly divided into three categories: LSTM based method, CNN based method and hybrid method.

our CNN-C with other state-of-the-art methods in Table III, where PE-Net indicates CNN-C with our position encoding scheme. Both Φ and Ψ is a 1D convolution operation with $1\times1\times16$ kernel, and Ω is the element-wise addition operation. To facilitate understanding on the differences among compared methods, the structures for different types of approaches are illustrated in Fig. 9. These methods fall into three types: the LSTM based method, CNN based method and hybrid method. As presented in Section II, these methods utilize different neural networks to exploit the temporal dependencies for the RUL prediction. For the CNN based method, CNN is used as a feature encoder to capture the temporal information. In the LSTM based method, the LSTM or Bi-LSTM is used to exploit the long-term memories. For the hybrid method, a CNN is firstly used to extract feature vectors. Then, an LSTM is used to capture the temporal information for the RUL prediction.

In these methods, Deep LSTM [15] utilizes LSTM to extract a feature map for the RUL prediction. Huang's BLSTM [17] adopts a bi-directional LSTM for the RUL prediction. Benefiting from bi-directional LSTM, Huang's BLSTM [17] performs better than Deep LSTM [15] on FD004. Chen's Hybrid method [1] is proposed based on a hybrid of CNN and LSTM. It achieves better performance than Deep LSTM [15] on FD001, and performs comparably to Deep LSTM [15] on FD004. Among these methods, KDnet [10] utilizes knowledge distillation to transfer knowledge in a well trained LSTM to a CNN. Although KDnet [10] achieves remarkable performance, its training process is complex and time-consuming. Compared with other methods, our proposed CNN-C achieves comparable performance without requiring a complex training process. After adding our proposed position encoding scheme, the performance of our PE-Net on nearly all subsets are further improved, which sets new state-of-the-art performances on *C-MAPSS* Dataset.

In Table III, we also compare our methods with three CNN based methods: Babu's CNN [14], Li's CNN [23] and KDnet [10]. It can be found that our methods perform better than other CNN based methods on nearly all subsets. Even on FD001, our PE-Net achieves the second best performance in terms of Score. These comparison results verify our analysis, showing our novel CNN's strong capability of capturing sequential information for the RUL prediction.

C. Analysis Experiments

In the subsection, we analyze our method performances

under different network configurations and different position encoding parameters. Since FD004 is the most complicated among four subsets in *C-MAPSS* Dataset, we choose to carry out experiments on FD004.

1) Analysis for Different Network Configurations: We believethatthereceptivefieldaffectstheCNN'ssequentialinformation capability. To verify this hypothesis, several experiments are carried out, which are listed in Table IV.

TABLE IV
THE PERFORMANCES OF OUR PROPOSED CNNS ON FD004

Network	Convolution unit no.	Receptive field	RMSE	Score
CNN-A	2	9	16.86	1324.96
CNN-B	4	21	17.28	1293.96
CNN-C	6	33	16.29	**1204.75**
CNN-D	7	49	**15.86**	1205.08
CNN-S1	6	19	18.99	1956.41
CNN-S2	2	31	16.39	1367.76
CNN-S3	6	121	17.84	1502.80

In Table IV , for each proposed CNN, its corresponding convolution unit number and receptive field on the final produced feature map are listed. Since the temporal length for the segmented input signal is 30, a CNN whose receptive field is larger than 30, can theoretically capture sequential information. As shown in Table IV , CNN-D performs similarly to CNN-C. It may be because CNN-C with a receptive field of 33 can fully exploit sequential information. Improvement by further increasing receptive field is limited. Among CNN-A, CNN-B and CNN-C, the performance of the CNN is improved when its corresponding receptive field is enlarged. This verifies our analysis on the relationship between the CNN's receptive field and the sequential information capacity.

Since the performances of CNN may be affected by other factors, we conduct another three comparison experiments (CNN-S1, CNN-S2 and CNN-S3) to further verify our hypothesis. To distinguish CNN-S variants with CNN-C, we illustrate these four networks in Fig. 10 , where the input tensor, the produced feature maps and the receptive fields for the last convolutional layer are shown. The major differences between CNN-S variants and CNN-C are highlighted in red and pink. The depth of CNN may also affect its performance. To avoid the influence of the CNN's depth, an experiment called CNN-S1 is conducted. In CNN-S1, we only change the stride value for the first convolutional layer from 2 to 1 and preserve other configurations. As shown in Fig. 10 , after changing the network configuration, the receptive field of CNN-S1 is reduced. According to Table IV , the depth of CNN-S1 which is represented as a convolution unit number, is the same as that of CNN-C, while the receptive field decreases from 33 to 19. The performance of CNN-S1 is inferior to that of CNN-C. This demonstrates that the receptive filed is important for the CNN to capture the sequential information, which further verifies our hypothesis in Section III-A-2).

To enlarge the receptive field of the CNN, an alternative method is to apply a convolutional kernel with a large size. However, it is difficult to optimize this large kernel, which may degrade the performance of CNN. To verify our assumption, we conduct another experiment denoted as CNN-S2. CNN-S2 consists of two convolution units, where these two convolutional layer parameters are 15×1 with channel 16 and stride 4, and 5×1 with channel 64 and stride 5, respectively. For the decoder component, CNN-S2 adopts the same parameters as others', where two FC layers with 256 hidden number are used. In Fig. 10, it can be found that, the receptive field of CNN-S2 is nearly unchanged, while the network depth is decreased from 6 to 2. As listed in Table IV , although the receptive field for CNN-S2 is similar to that of CNN-C, CNN-S2 performs worse than CNN-C. This supports our hypothesis in Section III-A-3) and demonstrates that the kernel size of CNN should be small.

Apart from the receptive field and kernel size, the channel number may also affect the performance of a CNN. To verify our hypothesis, we conduct another experiment called CNN-S3. In CNN-S3, all kernels are set as 9×1 and the variation of channel number is opposite to that in CNN-C, which decreases from 512 to 64. As shown in Fig. 10 , the feature depth in CNN-S3 gradually decreases, which is opposite to that in other networks. The receptive field in CNN-S3 is very enlarged and is even larger than the size of input signals. From Table IV, it can be found that though the receptive field for CNN-S3 is much larger than that of CNN-C, CNN-S3 is still surpassed by CNN-C. This indicates that the configuration of the channel number is important for a CNN to capture sequential information, and the channel number should increase gradually, which also verifies our conclusion in Section III-B.

2) Analysis for Position Encoding Scheme: In this part, we conduct experiments to analyze our method performances under different position encoding parameters.

Parameter p_d: Position encoding dimension p_d determines the size of the position encoding vector. We conduct three experiments with different p_d values, where the Ψ is a $1\times3\times16$ convolutional layer and Φ is a $1\times1\times16$ convolutional layer. As shown in Fig. 11 , our method performs the best (15.83 in RMSE and 1153.18 in Score), when p_d is set as 16. When we increase the position encoding dimension, our method shows relatively stable performances in Score.

Parameter Ψ: Ψ is the transformation for input signals. As shown in Fig. 12, three experiments are conducted to show the impact of the Ψ parameter on performance, where p_d is set as 16, and Φ is a $1\times1\times16$ convolutional layer. When we increase the kernel size in Φ, performance decreases. This indicates that the context information among the input signal may affect the encoded position information. Our method performs the best with the $1\times1\times16$ convolutional layer.

Parameter Φ: Φ is the transformation for position encoding information. In Table V , two experiments are conducted. A $1\times1\times16$ convolutional layer is firstly used to transform the position encoding information. In comparison, we remove the convolutional layer and directly fuse the position encoding information with the transformed input signals, where Ψ is a $1\times1\times16$ convolutional layer. From Table V , it demonstrates that our proposed transformation Φ can improve the RUL

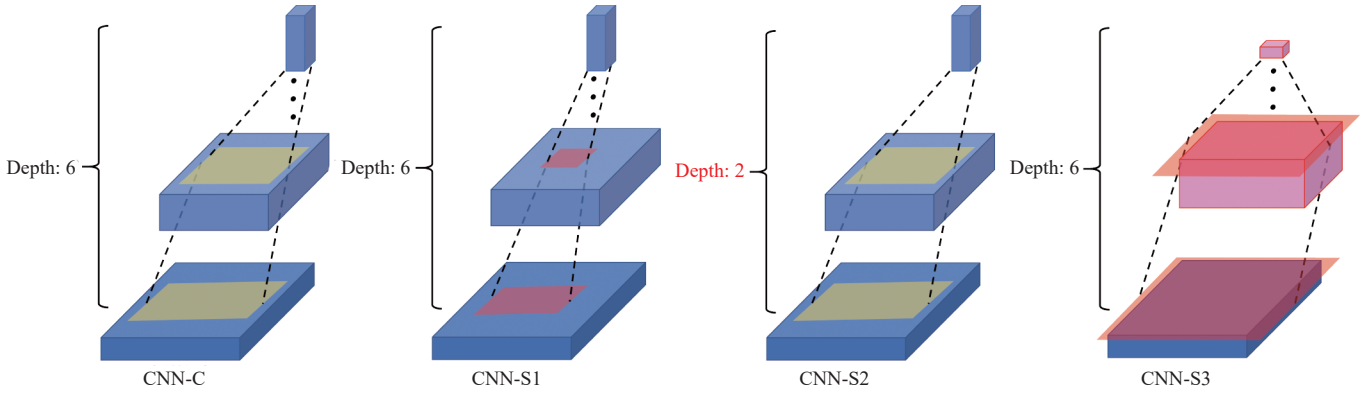


Fig. 10. Comparison of CNN-C and CNN-S variants. The input, the produced feature maps and the corresponding receptive field are illustrated for these four neural networks.

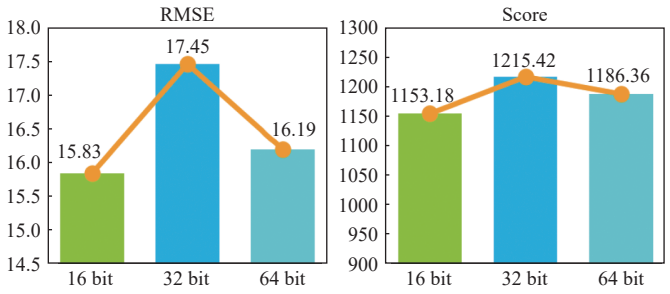


Fig. 11. Results of parameter p_d analysis experiments. It can be found that our method achieves the best performance when p_d is 16.

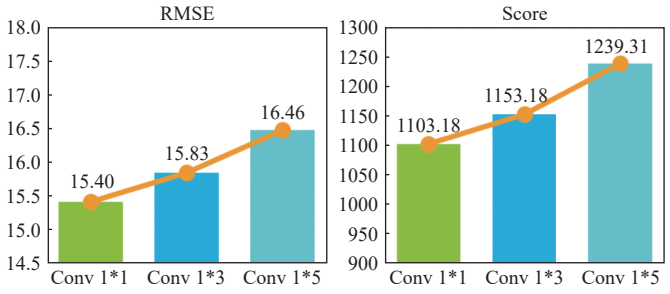


Fig. 12. Results of parameter Ψ analysis experiments. Our approach shows the best performance with the $1\times1\times16$ convolutional layer.

TABLE V
ANALYSIS EXPERIMENTS FOR Φ

Φ parameter	RMSE	Score
conv $1\times1\times16$	**15.40**	**1103.18**
N.A.	16.61	1194.87

prediction accuracy.

Fusion Method Ω*:* We propose the fusion method Ω to combine the transformed input signals and position encoding information. In this part, we compare two fusion methods: element-wise addition fusion and concatenation fusion in Table VI. It can be seen that the performances for the addition fusion are better than that for the concatenation fusion. This indicates that element-wise addition fusion can more effectively fuse position encoding information with input signals.

TABLE VI
ANALYSIS EXPERIMENTS FOR FUSION METHOD

Fusion method	RMSE	Score
Addition	**15.40**	**1103.18**
Concatenation	15.80	1139.73

D. Prediction Encoding Influence on RNN

Our proposed position encoding scheme has shown effectiveness on the RUL prediction in our PE-Net. In this subsection, we investigate whether the position encoding scheme can benefit the RNN based method.

We firstly implement a RNN based method according to [17], where Bi-LSTM is used to learn the long-term dependencies. After that, we apply our proposed position encoding scheme on this RNN based method, which is denoted as RNN+PE. The corresponding experimental results are listed in Table VII. As shown in Table VII, RNN+PE performs better than RNN, indicating that our proposed position encoding scheme is also effective for RNN based methods. Although this Bi-LSTM based method is improved with our position encoding scheme, it is still inferior to our PE-Net. This may be because our novel CNN in the PE-Net can provide high-qualified feature representations for the RUL prediction. Overall, our proposed position encoding scheme is effective for both RNN and CNN based methods.

E. Prediction Result Analysis on Test Data

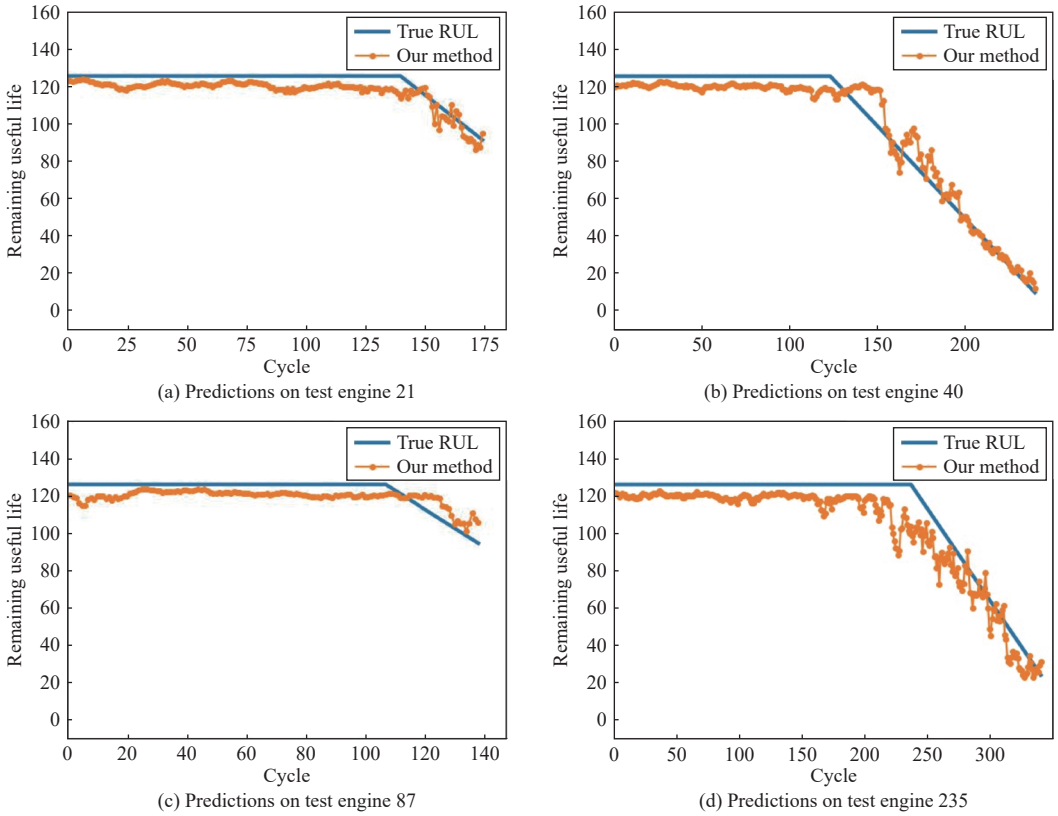
In the *C-MAPSS* dataset [39], the test trajectories contain various scenarios. To analyze our proposed PE-Net performance on different scenarios, we show our PE-Net predictions on the FD004 subset in this subsection.

We firstly illustrate four examples of life-time RUL predictions on the test engines. As listed in Fig. 13, four test engines with different RUL degradation characteristics are listed. In Figs. 13(a) and 13(c), the true RUL is near the predefined maximum RUL, and our predictions are very close to the true RUL. From Figs. 13(b) and 13(d), it can be seen that the true RUL varies largely, and scenarios become complex. Our PE-Net can still accurately predict the true RUL.

We also visualize the prediction results of all test engines on the FD004 subset in Fig. 14. It can be found that our PE-Net is

TABLE VII
ANALYSIS EXPERIMENTS ON THE POSITION ENCODING SCHEME FOR RNN

Dataset	FD001		FD002		FD003		FD004	
Evaluation	RMSE	Score	RMSE	Score	RMSE	Score	RMSE	Score
RNN (ours)	**13.33**	309.56	16.63	1681.04	14.92	400.92	17.00	1404.40
RNN+PE (ours)	14.26	288.19	15.68	1427.29	14.55	305.57	16.72	1127.50
PE-Net (ours)	13.98	**280.87**	**14.69**	**881.73**	**12.33**	**272.85**	**15.40**	**1103.18**



(a) Predictions on test engine 21
(b) Predictions on test engine 40
(c) Predictions on test engine 87
(d) Predictions on test engine 235

Fig. 13. Illustration for the RUL predictions on four test engines. It can be found that our PE-Net is able to predict the RUL accurately in both simple and complex scenarios.

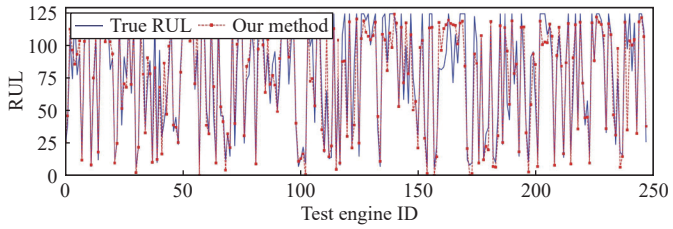


Fig. 14. Our PE-Net predictions on FD004. This shows that our PE-Net can predict the RUL accurately.

able to predict RUL accurately, which demonstrates that our proposed PE-Net is effective.

V. CONCLUSION AND FUTURE WORK

In this paper, we have investigated the relationship between CNN's architecture and the capability for sequential information. Based on our analysis, we have proposed a series of CNNs, whose capability for sequential information has been improved. Extensive experiments have been carried out. According to our experimental results, CNN based approaches are able to perform comparably to RNN based methods and hybrid based methods. Additionally, we have proposed a position encoding scheme to further improve the sequential information capability. Experimental results have shown that our proposed position encoding scheme effectively improves the CNN's performance. With our proposed position encoding scheme, our PE-Net surpasses nearly all methods and sets new state-of-the-art performance on the *C-MAPSS* dataset.

The inputs for the RUL prediction are generally taken from multiple sensor data, which can be regarded as a high-dimensional and sparse (HiDS) matrix. Our current method, PE-Net, directly processes input signals, which is time-consuming and redundant. In the future, for dealing with this data more efficiently, we will investigate how to apply technologies [42]–[44] related to the HiDS matrix processing for efficient processing of high-dimensional data.

REFERENCES

[1] Z. H. Chen, M. Wu, R. Zhao, F. Guretno, R. Q. Yan, and X. L. Li,

"Machine remaining useful life prediction via an attention-based deep learning approach," *IEEE Trans. Ind. Electron.*, vol. 68, no. 3, pp. 2521–2531, Mar. 2021.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learning Representations*, San Diego, USA, 2015.

[3] K. M. He, X. Y. Zhang, S. Q. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Las Vegas, USA, 2016, pp. 770–778.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. 31st Int. Conf. Neural Information Processing Systems*, Long Beach, USA, 2017, pp. 6000–6010.

[5] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, Minnesota, 2018, pp. 4171–4186.

[6] A. T. Khan, S. Li, and X. W. Cao, "Human guided cooperative robotic agents in smart home using beetle antennae search," *Sci. China Inf. Sci.*, vol. 65, no. 2, Article No. 122204, Jan. 2022.

[7] A. T. Khan, X. W. Cao, Z. Li, and S. Li, "Enhanced beetle antennae search with zeroing neural network for online solution of constrained optimization," *Neurocomputing*, vol. 447, pp. 294–306, Aug. 2021.

[8] A. T. Khan, S. Li, and Z. Li, "Obstacle avoidance and model-free tracking control for home automation using bio-inspired approach," *Adv. Control Appl.*, vol. 4, p. e63, Dec. 2021.

[9] J. D. Lin, Z. Lin, G. B. Liao, and H. P. Yin, "A novel product remaining useful life prediction approach considering fault effects," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 11, pp. 1762–1773, Nov. 2021.

[10] Q. Xu, Z. H. Chen, K. Y. Wu, C. Wang, M. Wu, and X. L. Li, "KDnet-RUL: A knowledge distillation framework to compress deep neural networks for machine remaining useful life prediction," *IEEE Trans. Ind. Electron.*, vol. 69, no. 2, pp. 2022–2032, Feb. 2022.

[11] Y. Qin, D. L. Chen, S. Xiang, and C. C. Zhu, "Gated dual attention unit neural networks for remaining useful life prediction of rolling bearings," *IEEE Trans. Ind. Inf.*, vol. 17, no. 9, pp. 6438–6447, Sep. 2021.

[12] X. W. Zhang, Y. Qin, C. Yuen, L. Jayasinghe, and X. Liu, "Time-series regeneration with convolutional recurrent generative adversarial network for remaining useful life estimation," *IEEE Trans. Ind. Inf.*, vol. 17, no. 10, pp. 6820–6831, Oct. 2021.

[13] B. Y. Yang, R. N. Liu, and E. Zio, "Remaining useful life prediction based on a double-convolutional neural network architecture," *IEEE Trans. Ind. Electron.*, vol. 66, no. 12, pp. 9521–9530, Dec. 2019.

[14] G. S. Babu, P. L. Zhao, and X. L. Li, "Deep convolutional neural network based regression approach for estimation of remaining useful life," in *Proc. 21st Int. Conf. Database Systems for Advanced Applications*, Dallas, USA, 2016, pp. 214–228.

[15] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, "Long short-term memory network for remaining useful life estimation," in *Proc. IEEE Int. Conf. Prognostics and Health Management*, Dallas, USA, 2017, pp. 88–95.

[16] J. J. Zhang, P. Wang, R. Q. Yan, and R. X. Gao, "Long short-term memory for machine remaining life prediction," *J. Manuf. Syst.*, vol. 48, pp. 78–86, Jul. 2018.

[17] C. G. Huang, H. Z. Huang, and Y. F. Li, "A bidirectional LSTM prognostics method under multiple operational conditions," *IEEE Trans. Ind. Electron.*, vol. 66, no. 11, pp. 8792–8802, Nov. 2019.

[18] L. Ren, J. B. Dong, X. K. Wang, Z. H. Meng, L. Zhao, and M. J. Deen, "A data-driven auto-CNN-LSTM prediction model for Lithium-Ion battery remaining useful life," *IEEE Trans. Ind. Inf.*, vol. 17, no. 5, pp. 3478–3487, May 2021.

[19] L. Jayasinghe, T. Samarasinghe, C. Yuenv, J. C. N. Low, and S. S. Ge, "Temporal convolutional memory networks for remaining useful life estimation of industrial machinery," in *Proc. IEEE Int. Conf. Industrial Technology*, Melbourne, Australia, 2019, pp. 915–920.

[20] M. Xia, T. Li, T. X. Shu, J. F. Wan, C. W. De Silva, and Z. R. Wang, "A two-stage approach for the remaining useful life prediction of bearings using deep neural networks," *IEEE Trans. Ind. Inf.*, vol. 15, no. 6, pp. 3703–3711, Jun. 2019.

[21] R. H. Jiao, K. X. Peng, and J. Dong, "Remaining useful life prediction for a roller in a hot strip mill based on deep recurrent neural networks," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 7, pp. 1345–1354, Jul. 2021.

[22] C. Chen, N. Y. Lu, B. Jiang, and C. S. Wang, "A risk-averse remaining useful life estimation for predictive maintenance," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 2, pp. 412–422, Feb. 2021.

[23] X. Li, Q. Ding, and J. Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Reliab. Eng. Syst. Saf.*, vol. 172, pp. 1–11, Apr. 2018.

[24] M. Ragab, Z. H. Chen, M. Wu, C. K. Kwoh, R. Q. Yan, and X. L. Li, "Attention-based sequence to sequence model for machine remaining useful life prediction," *Neurocomputing*, vol. 466, pp. 58–68, Nov. 2021.

[25] J. Y. Wu, M. Wu, Z. H. Chen, X. L. Li, and R. Q. Yan, "Degradation-aware remaining useful life prediction with LSTM autoencoder," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–10, Feb. 2021.

[26] C. Szegedy, W. Liu, Y. Q. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Boston, USA, 2015, pp. 1–9.

[27] A. G. Howard, M. L. Zhu, B. Chen, D. Kalenichenko, W. J. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv: 1704.04861, Apr. 2017.

[28] D. Wang, F. F. Yang, K. L. Tsui, Q. Zhou, and S. J. Bae, "Remaining useful life prediction of Lithium-Ion batteries based on spherical cubature particle filter," *IEEE Trans. Instrum. Meas.*, vol. 65, no. 6, pp. 1282–1291, Jun. 2016.

[29] M. Jouin, R. Gouriveau, D. Hissel, M. C. Péra, and N. Zerhouni, "Particle filter-based prognostics: Review, discussion and perspectives," *Mech. Syst. Signal Proc.*, vol. 72–73, pp. 2–31, May 2016.

[30] A. Soualhi, H. Razik, G. Clerc, and D. D. Doan, "Prognosis of bearing failures using hidden Markov models and the adaptive neuro-fuzzy inference system," *IEEE Trans. Ind. Electron.*, vol. 61, no. 6, pp. 2864–2874, Jun. 2014.

[31] R. Khelif, B. Chebel-Morello, S. Malinowski, E. Laajili, F. Fnaiech, and N. Zerhouni, "Direct remaining useful life estimation based on support vector regression," *IEEE Trans. Ind. Electron.*, vol. 64, no. 3, pp. 2276–2285, Mar. 2017.

[32] Z. G. Tian, "An artificial neural network method for remaining useful life prediction of equipment subject to condition monitoring," *J. Intell. Manuf.*, vol. 23, no. 2, pp. 227–237, Apr. 2012.

[33] L. Saidi, J. Ben Ali, E. Bechhoefer, and M. Benbouzid, "Wind turbine high-speed shaft bearings health prognosis through a spectral kurtosis-derived indices and SVR," *Appl. Acoust.*, vol. 120, pp. 1–8, Mar. 2017.

[34] J. Carreira and A. Zisserman, "Quo Vadis, action recognition? A new model and the kinetics dataset," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Honolulu, USA, 2017, pp. 4724–4733.

[35] X. Z. Zhu, Y. J. Wang, J. F. Dai, L. Yuan, and Y. C. Wei, "Flow-guided feature aggregation for video object detection," in *Proc. IEEE Int. Conf. Computer Vision*, Venice, Italy, 2017, pp. 408–417.

[36] R. B. Jin, G. S. Lin, C. Y. Wen, J. L. Wang, and F. Y. Liu, "Feature flow: In-network feature flow estimation for video object detection," *Pattern Recogn.*, vol. 122, p. 108323, Feb. 2022. DOI: 10.1016/j.patcog.2021.108323.

[37] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. 13th European Conf. Computer Vision*, Zurich, Switzerland, 2014, pp. 818–833.

[38] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Machine Learning*, Lille, France, 2015, pp. 448–456.

[39] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *Proc. Int. Conf. Prognostics and Health Management*, Denver, USA, 2008, pp. 1–9.

[40] S. Behera and R. Misra, "Generative adversarial networks based remaining useful life estimation for IIoT," *Comput. Electr. Eng.*, vol. 92, p. 107195, Jun. 2021. DOI: 10.1016/j.compeleceng.2021.107195.

[41] C. Zhang, P. Lim, A. K. Qin, and K. C. Tan, "Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2306–2318, Oct. 2017.

[42] H. Y. Lu, L. Jin, X. Luo, B. L. Liao, D. S. Guo, and L. Xiao, "RNN for solving perturbed time-varying underdetermined linear system with double bound limits on residual errors and state variables," *IEEE Trans. Ind. Inf.*, vol. 15, no. 11, pp. 5931–5942, Nov. 2019.

[43] L. Xin, Y. Yuan, M. C. Zhou, Z. G. Liu, and M. S. Shang, "Non-negative latent factor model based on β-divergence for recommender systems," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 51, no. 8, pp. 4612–4623, Aug. 2021.

[44] X. Luo, Z. D. Wang, and M. S. Shang, " An instance-frequency-weighted regularization scheme for non-negative latent factor analysis on high-dimensional and sparse data," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 51, no. 6, pp. 3522–3532, Jun. 2021.

Ruibing Jin received the B.Eng. degree from the University of Electronic Science and Technology of China in 2014 and the M.Eng and the Ph.D. degrees from Nanyang Technological University, Singapore in 2016 and 2020, respectively. He is a Scientist at Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore. He won the First Place Winner in the CVPR 2021 UG2+ Challenge. His research interests include computer vision, machine learning, time series and related applications.

Min Wu (Senior Member, IEEE) is currently a Senior Scientist in Data Analytics Department, Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore. He received the Ph.D. degree in computer science from Nanyang Technological University (NTU), Singapore, in 2011 and B.S. degree in computer science from University of Science and Technology of China (USTC) in 2006. He received the best paper awards in InCoB 2016 and DASFAA 2015. He also won the IJCAI competition on repeated buyers prediction in 2015. His current research interests include machine learning, data mining and bioinformatics.

Keyu Wu received the B.Eng degree in bioengineering from National University of Singapore, Singapore, in 2013 and the Ph.D. degree in electrical and electronic engineering from Nanyang Technological University, Singapore, in 2020. She is currently a Scientist with the Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore. Her research interests include reinforcement learning, transfer learning, self-supervised learning, and related applications.

Kaizhou Gao (Member, IEEE) received the B.Sc. and masters degrees from Liaocheng University and Yangzhou University, in 2005 and 2008, respectively, and the Ph.D. degree from Nanyang Technological University (NTU), Singapore, in 2016. From 2008 to 2012, he was with the School of Computer, Liaocheng University. From 2012 to 2013, he was a Research Associate with the School of Electronic and Electrical Engineering, NTU, where he has been a Research Fellow from 2015 to 2018. He is currently an Assistant Professor with the Macau Institute of Systems Engineering, Macau University of Science and Technology, China. His research interests include intelligent computation, optimization, scheduling, and intelligent transportation. He has published over 80 refereed papers. He is an Associate Editor of *International Journal Swarm and Evolutionary Computation*.

Zhenghua Chen (Senior Member, IEEE) received the B.Eng. degree in mechatronics engineering from University of Electronic Science and Technology of China (UESTC) in 2011, and the Ph.D. degree in electrical and electronic engineering from Nanyang Technological University (NTU), Singapore, in 2017. He has been working at NTU as a Research Fellow. Currently, he is a Scientist and Lab Head at Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore. He has won several competitive awards, such as First Place Winner for CVPR 2021 UG2+ Challenge, A*STAR Career Development Award, First Runner-Up Award for Grand Challenge at IEEE VCIP 2020, Finalist Academic Paper Award at IEEE ICPHM 2020, etc. He serves as Associate Editor for Elsevier Neurocomputing and Guest Editors for five journals. He is currently the Vice Chair of IEEE Sensors Council Singapore Chapter and IEEE Senior Member. His research interests include smart sensing, data analytics, machine learning, transfer learning and related applications.

Xiaoli Li (Senior Member, IEEE) is currently a Principal Scientist at the Institute for Infocomm Research, A*STAR, Singapore. He also holds adjunct Full Professor position at Nanyang Technological University. His research interests include data mining, machine learning, AI, and bioinformatics. He has been serving as area chairs/senior PC members in leading AI and data mining related conferences (including IJCAI, AAAI, KDD, ICDM). He is currently serving as Editor-in-Chief of *World Scientific Annual Review of Artificial Intelligence*, and Associate Editor of *IEEE Transactions on Artificial Intelligence* and *Machine Learning with Applications* (Elsevier). He has published more than 260 high quality papers and won 8 best paper/benchmark competition awards.