Multivariate Time Series Representation Learning via Hierarchical Correlation Pooling Boosted Graph Neural Network

Yucheng Wang, Min Wu, Xiaoli Li, Lihua Xie, *Fellow, IEEE*, and Zhenghua Chen

***Abstract*—Representation learning is vital for the performance of Multivariate Time Series (MTS) related tasks. Given high-dimensional MTS data, researchers generally rely on deep learning (DL) models to learn representative features. Among them, the methods that can capture the spatial-temporal dependencies within MTS data generally achieve better performance. However, they ignored hierarchical relations and the dynamic property within MTS data, hindering their performance. To address these problems, we propose a Hierarchical Correlation Pooling boosted graph neural network (HierCorrPool) for MTS data representation learning. First, we propose a novel correlation pooling scheme to learn and capture hierarchical correlations between sensors. Meanwhile, a new assignment matrix is designed to ensure the effective learning of hierarchical correlations by adaptively combining both sensor features and correlations. Second, we learn sequential graphs to represent the dynamic property within MTS data, so that this property can be captured for learning decent representations. We conducted extensive experiments to test our model on various MTS tasks, including remaining useful life prediction, human activity recognition, and sleep stage classification. Experimental results have shown the effectiveness of our proposed model.**

***Impact Statement*—Multivariate Time Series (MTS) data representation learning has received great attention in recent years due to its importance for downstream tasks. MTS has two important properties, the temporal and spatial dependencies. Traditionally, due to highly nonlinear power, deep learning based methods have been widely applied for addressing the former, but few works focus on making full of the spatial dependencies. Although pioneers started to exploit Graph Neural Network to capture the spatial dependencies, they still have limitations. The proposed method in this paper learns and captures the hierarchical correlations between sensors, and meanwhile, sequential graphs are learned to represent the dynamic property within MTS data. In this way, the spatial-temporal dependencies within MTS data can be better leveraged. With significant improvements on multiple MTS tasks compared to state-of-the-art algorithms, our method is ready to learn decent representations from MTS signals in different areas.**

***Index Terms*—Feature Representation Learning, Graph Neural Network (GNN), Multivariate Time Series (MTS) Data**

This research is supported by the Agency for Science, Technology and Research (A*STAR) under its Career Development Award (Grant No. C210112046) (*Corresponding Author: Zhenghua Chen*).

Yucheng Wang is with Institute for Infocomm Research, A*STAR, Singapore and the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore (Email: yucheng003@e.ntu.edu.sg).

Min Wu is with Institute for Infocomm Research, A*STAR, Singapore (Email: wumin@i2r.a-star.edu.sg).

Lihua Xie is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore (Email: elhxie@ntu.edu.sg).

Xiaoli Li and Zhenghua Chen are with Institute for Infocomm Research and Centre for Frontier AI Research, A*STAR, Singapore (Email: xlli@i2r.a-star.edu.sg, chen0832@e.ntu.edu.sg).

I. Introduction

Multivariate time series (MTS) data refer to the sequential signals ordered in time and collected from multiple sources (e.g., sensors). The historical information and the relations between sources should be analyzed for various downstream tasks, such as prediction [1], [2], e.g., machine Remaining Useful Life (RUL) prediction, and classification [3], [4], e.g., Human Activity Recognition (HAR) and Sleep Stage Classification (SSC). Traditionally, researchers adopted model-based methods [5], statistic-based methods [6], distance-based methods [7], and feature-based methods [8] to process MTS data. However, these traditional methods require expert knowledge and are thus labour-intensive. Besides, spatial (i.e., the relations between sources) and temporal information (i.e., the temporal dependencies among historical information) within MTS data should be analyzed simultaneously, but it is hard for the traditional models to capture such information.

To achieve satisfactory performance for downstream tasks, learning decent representations from MTS data became popular in recent years [8], [9]. Among them, Deep Learning (DL)-based models have shown to be the leading solutions [10]. Due to the capability of nonlinear modeling and automatic feature extraction, DL-based models can learn informative features without explicitly knowing the complicated dynamics of MTS data. Thus, DL-based models, such as Convolutional Neural Network (CNN) [11], [12], [13] and Long Short-Term Memory (LSTM) [14], [15], [16], have been widely adopted to learn representations of MTS data. These models have been shown to perform well in capturing temporal information of time-series data, thus achieving better performance than traditional models. However, they are incapable of well capturing spatial dependencies, i.e., the correlations between sensors, hindering their performance in representation learning.

Graph Neural Network (GNN) serves as a good solution to the above problem because of its great success in modelling structured data. Recently, many researchers have started exploiting GNN to learn MTS representations by capturing sensor correlations. These GNN-based models have shown validity in various MTS related areas, such as RUL prediction [17], SSC [18], and HAR [3]. Although they have achieved better performance than traditional DL-based models, they still have limitations in fully utilizing the correlations between sensors. As shown in Fig. 1 (a.1), most works [19], [20] directly stack the information of sensors to learn representations after updating sensor features by capturing correlations with GNN.

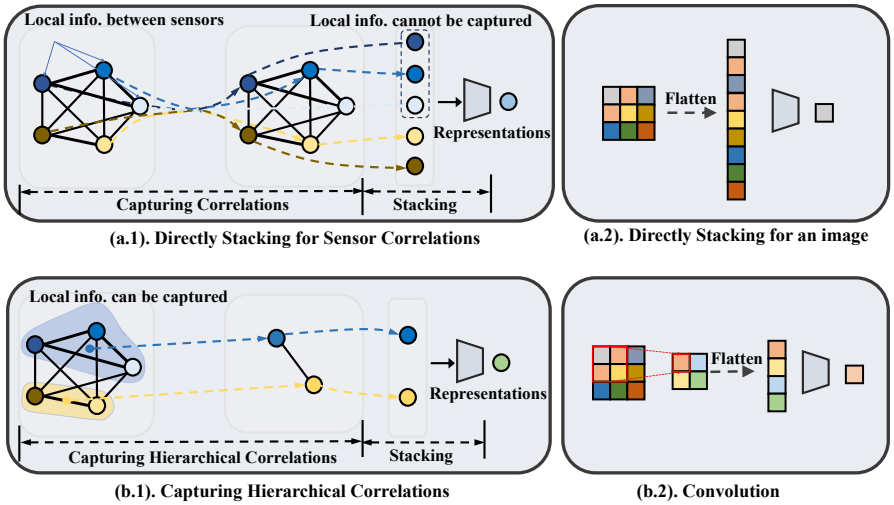

Fig. 1: The diagram of differences between (a) directly stacking and (b) capturing hierarchical correlations.

However, the way of direct stacking ignores the important local information between sensors, while it is more reasonable to first analyze the local and then the overall spatial information to learn representations. This is similar with Computer Vision (CV). For example, researchers in CV prefer to use CNN as Fig. 1 (b.2) instead of directly stacking all pixels to extract features from images, as the latter cannot capture the local spatial information and also requires more trainable parameters as shown in Fig. 1 (a.2). Similarly, the lack of local information between sensors hinders the performance of MTS representation learning. Besides, it is noted that MTS data show dynamic sensor correlations, i.e., the correlations between sensors are dynamically changing over time. Let's take machine RUL prediction as an example. The correlation between a fan speed sensor and a temperature sensor will be different when a machine is in different states, i.e., in good or bad condition. To achieve better representation learning, it is useful to capture this dynamic property when generating the graph structure for MTS data. However, current methods [19], [20] mainly construct one graph for each MTS sample and are thus difficult to leverage the dynamic sensor correlations.

To fully leverage the correlations between sensors, we propose a Hierarchical Correlation Pooling boosted graph neural network (HierCorrPool). We consider the local correlations as functional units. Intuitively, we should exploit the local inter-correlations of sensors at first and then the overall correlations when analyzing the MTS data, as shown in Fig. 1 (b.1). To analyze the local correlations, we learn and capture hierarchical correlations between sensors layer by layer. With the local information being captured, we are able to learn better representations. To achieve good performance, it is necessary to learn decent hierarchical correlations. Traditionally, graph pooling methods, such as DiffPool [21], were proposed to handle binary graphs and learn hierarchical graphs by considering node features. However, MTS data will derive a weighted graph instead of a binary graph, so the traditional graph pooling methods may not perform well in MTS related areas. To address this problem, we propose a novel correlation pooling scheme to learn the hierarchical correlations between sensors. Specifically, to incorporate the weighted sensor correlations into learning hierarchical correlations, we propose to learn a new assignment matrix by combining the information from sensor correlations and features adaptively, which is then used to learn precise hierarchical correlations. Besides, we design and learn sequential graphs for representing the dynamic sensor correlations within MTS data. To achieve this, we first divide an MTS sample into multiple feature windows, and the sequential feature windows show temporal dependencies. Then, we adopt CNN to capture the temporal dependencies. With the learned windows, we construct sequential graphs, which represent the dynamic sensor correlations and can be then utilized to learn and capture the hierarchical correlations by our correlation module and GNN respectively. In this way, the dynamic sensor correlations can be leveraged to learn decent representations for MTS data.

Our contributions can be summarized as follows:

- To ensure effective learning and capturing of hierarchical correlations between sensors, a novel correlation pooling scheme is proposed. Specifically, a new assignment matrix is designed by adaptively combining the information from both sensor correlations and features.
- To represent the dynamic sensor correlations within MTS data, sequential graphs are designed and learned. Then, the hierarchical correlations in the sequential graphs are leveraged to learn representative features for MTS data.
- To evaluate the effectiveness of our method, we conduct extensive experiments on different MTS tasks. The results demonstrate that our method outperforms state-of-the-art methods on various MTS tasks.

II. Related Work

This section reviews the related work on DL-based representation learning according to different model types.

A. CNN-based Models for MTS Representation Learning

Due to the popularity in computer vision, CNN-based models have been widely used for learning the representations of

MTS data. To capture the temporal dependencies in time-series data, 1D CNN was initially applied. Liu et al. [22] proposed a multivariate CNN by considering the multivariate and lag features for MTS classification. Wang et al. [23] proposed multiple CNNs to process the periodic information to make predictions based on MTS data. Besides, some researchers exploited 2D CNN to learn representations. Yang et al. [24] proposed to encode MTS data as two-dimensional images, which are concatenated as a bigger image and then processed by CNN to learn feature representation for classification.

B. LSTM-based Models for MTS Representation Learning

LSTM-based model is another branch to learn representations of MTS data by capturing the temporal dependencies. Du et al. [25] proposed a bi-directional-LSTM based temporal attention encoder-decoder model for MTS forecasting. Lu et al. [1] proposed a generative adversarial network combining LSTM and auto-encoder to learn representations for RUL prediction. Chen et al. [16] proposed an LSTM-based framework by combining attention to learn the importance of time steps for MTS prediction. These models can achieve better performance than those traditional models, showing the validity of DL-based solutions. However, most of these CNN- and LSTM-based models focus on the temporal dependencies of MTS data but ignore the correlations between sensors, which limits their performance in representation learning.

C. GNN-based Models for MTS Representation Learning

A few works have been devoted to leverage sensor correlations by GNN for MTS data. They mostly constructed a graph for each MTS sample and updated sensor features by leveraging sensor correlations with GNN. With the learned sensor features, they directly stacked the sensor features to learn overall representations [26], [27], [28]. For example, Ailin and Bryan [26] captured the correlations between sensors by GNN for anomaly detection. In the last layer of GNN, they stacked all sensor features and learned the representations by a neural network. Jia et al. [27] modeled the correlations between sensors by GNN for sleep stage classification. Then, they concatenated sensor features to learn the representations for classification. These methods are great pioneers. However, they cannot capture the hierarchical correlations between sensors. Meanwhile, they ignored the dynamic property of MTS data. These limitations restrict their performance.

To address the above problems, we learn the hierarchical correlations between sensors by proposing a novel correlation pooling module, and meanwhile, we learn sequential graphs to represent and capture the dynamic property within MTS data.

III. Hierarchical Correlation Pooling Boosted Graph Neural Network

A. Overall Structure

The overall structure of our proposed HierCorrPool is shown in Fig. 2. First, we divide an MTS sample into multiple feature windows, and the sequential feature windows show temporal dependencies, which are then captured by a 1D-CNN module. With the learned windows, we can then construct sequential graphs to represent the dynamic sensor correlations. The hierarchical correlations between sensors in each graph are learned by our proposed correlation pooling scheme. Then, GNN is adopted for each graph to learn sensor features by capturing the hierarchical correlations. Multiple layers of correlation pooling schemes are deployed to improve the nonlinear expressiveness and learn hierarchical correlations. Finally, after capturing the hierarchical correlations between sensors in sequential graphs, the learned hierarchical features are mapped to learn overall representations. The details of these modules are introduced in the following parts.

B. Sequential Graph Construction

It is noted that the correlations between sensors change dynamically. Let's take the fan speed sensor and the temperature sensor when predicting the RUL of a machine as an example. When the machine is in good condition, 50 r/s fan speed may only make the temperature go up to 50 degrees, but the same fan speed may cause higher temperature due to the bad condition of the machine, e.g., increased friction. The current works [19], [20] construct one graph for each MTS sample and thus cannot capture the dynamic sensor correlations. To represent the property, we here learn sequential graphs for each MTS sample.

Given an MTS sample $X = \{x_m\}_{m=1}^{N} \in \mathbb{R}^{N \times L}$ originating from N sensors, each sensor x_m has L timestamps, i.e., $\{x_m[1], x_m[2], ..., x_m[L]\}$. To learn decent sequential graphs, we first divide X into $\bar{L}$ small feature windows with fixed length f. Regarding the m-th sensor $x_m = \{x_m[t]\}_{t=1}^{L}$, for example, we divide it into $x_m = \{z_m[1], z_m[2], ..., z_m[\bar{L}]\}$ as shown in Fig. 3, where $z_m[1] = \{x_m[1], x_m[2], ..., x_m[f]\}$, $z_m[2] = \{x_m[f+1], x_m[f+2], ..., x_m[2f]\}$, etc. The sequential feature windows, i.e., $\{z_m[1], z_m[2], ..., z_m[\bar{L}]\}$, present temporal dependencies. To capture these temporal dependencies, we design a 1D-CNN module over the sequential feature windows. Besides, 1D-CNN has been proved to be capable of well capturing the temporal dependencies within MTS data in [29]. In particular, 1D-CNN consists of 1D convolutional kernels to capture the temporal dependencies. We suppose kernels with the learnable weights $W_C = \{w_C[1], w_C[2], ..., w_C[k]\}$, where k is the kernel size and $w_C[k] \in \mathbb{R}^{f \times f'}$. The kernels are convolved with $z_m = \{z_m[1], z_m[2], ..., z_m[\bar{L}]\}$ regarding the sensor v_m, as Eqn. (1).

$$z'_m = LeakyReLU(z_m * W_C). \tag{1}$$

Here, $*$ is the standard convolution operation, which is shown as $z'_m[t] = LeakyReLU(\sum_{\tau=1}^{k} z_m[t-\tau] w_C[\tau]), t \in [1, \bar{L}']$, where $\bar{L}'$ is the number of learned feature windows after convolution. Then, we can obtain the new features $z'_m \in \mathbb{R}^{\bar{L}' \times f'}$. As the 1D convolution has been done along the dimension of sequential feature windows, it is able to capture the temporal dependencies of feature windows.

In order to represent the spatial information (i.e., dynamic sensor correlations) within MTS data, we then design sequential graphs $\mathcal{G} = \{G[t]\}_{t=1}^{\bar{L}'}$, each of which contains node set V with features $Z'[t] \in \mathbb{R}^{N \times f'}$ and edges $E'[t] \in \mathbb{R}^{N \times N}$,

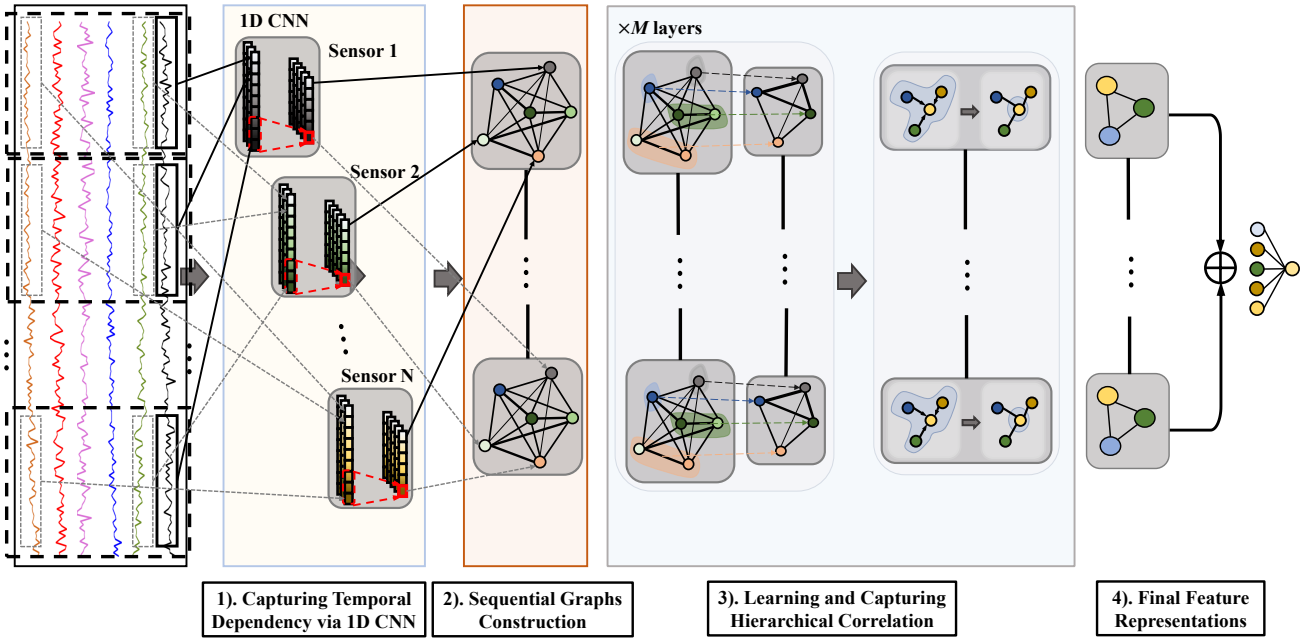

Fig. 2: The overall structure of our proposed HierCorrPool. 1). Each MTS sample is divided into multiple feature windows, and the sequential feature windows show temporal dependencies, which are captured by 1D-CNN; 2). Sequential graphs are constructed based on the learned windows, showing the changing dynamics of sensor correlations; 3). Hierarchical correlations for each graph are learned and captured by our correlation module and GNN respectively; 4). The learned sensor features are mapped to learn overall representations.

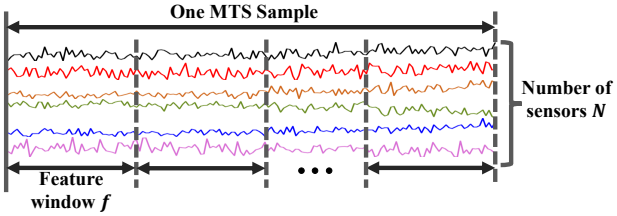

Fig. 3: The diagram of feature sampling.

representing sensor features and sensor correlations in this graph. For each graph, we already have sensor features $Z'[t] = \{z'_m[t]\}_{m=1}^{N}$, and we then need to construct edges $E'[t] = \{e'_{mn}[t]\}_{m,n=1}^{N}$ to represent the sensor correlations.

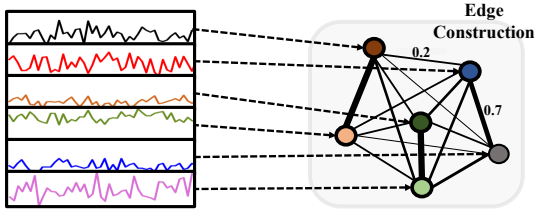

Fig. 4: The diagram of correlation construction.

We argue that the correlations between any two sensors are large when they have similar features, so we calculate the similarities between sensors as Fig. 4. Taking $z'_m[t]$ and $z'_n[t]$ as an example, the construction process is calculated as Eqn. (2), where the similarity is measured by the dot product of sensors, and $h(z'_m[t], z'_n[t]) = z'_m[t]z'_n[t]^T$ represents the correlation between sensor m and n. Furthermore, the softmax function is adopted to normalize the correlations, and thus the value of constructed $e'_{mn}[t]$ is within the range of [0, 1].

$$\begin{aligned} e'_{mn}[t] &= softmax(h(z'_m[t], z'_n[t])) \\ &= \frac{e^{h(z'_m[t], z'_n[t])}}{\sum_{i=1}^{N} e^{h(z'_m[t], z'_i[t])}} \in \mathbb{R}. \end{aligned} \tag{2}$$

By calculating the correlations of other sensor pairs, we obtain $E'[t]$ to represent the sensor correlations, i.e., spatial information, in the t-th graph. In this way, we obtain the sequential graphs to represent the dynamic changes of the sensor correlations (i.e., full spatial information). Then, the hierarchical correlations in each graph are learned and captured in the following parts.

C. Hierarchical Correlation Pooling Scheme

This part introduces the proposed correlation pooling scheme to learn the hierarchical correlations between sensors. As the sensors are with complex topological structures and they have no natural notion of spatial locality, the correlation pooling scheme is designed to handle the topological information between sensors.

To learn the hierarchical correlations between sensors, we need to design an assignment matrix $S^l \in \mathbb{R}^{N^l \times N^{l+1}}$ at first, which can map N^l nodes (i.e., clusters of sensors) to N^{l+1} nodes as shown in Fig. 5. The assignment matrix can cluster the sensors with similar properties and then map the dense correlations into the coarse correlations. Therefore, the assignment matrix is the key component for learning effective hierarchical correlations. Here, S^l represents the assignment

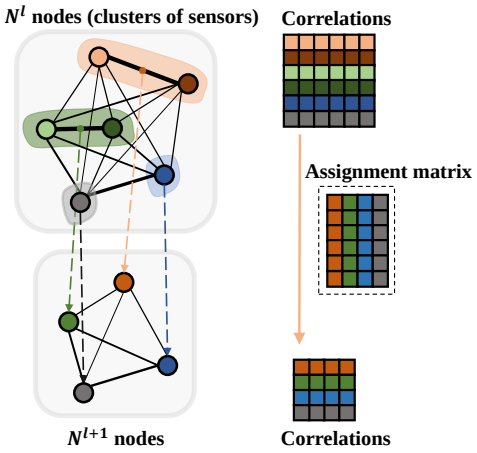

Fig. 5: The diagram of the learning of hierarchical correlations, where N^l nodes are mapped to N^{l+1} nodes by the assignment matrix. In this example, N^l and N^{l+1} are 6 and 4, respectively, and the assignment matrix is $S^l \in \mathbb{R}^{6\times 4}$.

matrix in the l-th layer, and the hierarchical correlations are learned layer by layer.

Traditionally, the assignment matrix is generated based solely on the node features for graph pooling, such as DiffPool [21], which was proposed mainly for the graphs with binary edges (an edge between any two nodes either exists or not). However, different from the binary graphs, the sensor correlation are weighted graphs as Eqn. (2). Thus, the edges of sensor correlations are more informative than those in binary graphs. Therefore, we are motivated to generate the assignment matrix by both the node features Z and edge weights E.

Given the t-th graph with node features as $Z^l[t] \in \mathbb{R}^{N^l\times d^l}$ and node correlations as $E^l[t] \in \mathbb{R}^{N^l\times N^l}$ in layer l, N^l and d^l represent the number of nodes and the feature dimension respectively. $Z^1[t]$ and $E^1[t]$ are $Z'[t]$ and $E'[t]$ respectively when $l = 1$. We aim to propose a scheme $\mathcal{S}$ to generate a precise assignment matrix $S^l[t]$ by considering both information from features and correlations, i.e., $S^l[t] = \mathcal{S}(Z^l[t], E^l[t]; W^l)$.

Concatenation: The first alternative scheme is the concatenation of node features and node correlations, as Eqn. (3).

$$S^l_C[t] = softmax(\sigma([E^l[t]||Z^l[t]]W^l_C)), \tag{3}$$

where $W_C \in \mathbb{R}^{(N^l+d^l)\times N^{l+1}}$. This scheme is a simple yet effective way to generate the assignment matrix, as it allows neural networks to decide automatically how many weights should be put on both parts. However, this scheme is only suitable for the situation where the scales of N^l and d^l are close. Otherwise, the neural network may bias towards the part with the larger size and thus have negative impacts on the performance of representation learning.

Concatenation + Projection: To address the mentioned problem, projection functions are introduced to map both parts into the same dimension, as Eqn. (4).

$$S^l_{CP}[t] = softmax(\sigma([\sigma(E^l[t]W^l_E)||\sigma(Z^l[t]W^l_Z)]W^l_{CP})), \tag{4}$$

where $W^l_E \in \mathbb{R}^{N^l\times H^l}$ and $W^l_Z \in \mathbb{R}^{d^l\times H^l}$ are projection functions, and $W^l_{CP} \in \mathbb{R}^{2H^l\times N^{l+1}}$. Here, H^l is the target dimension for both parts. With the projection functions, the scheme can generate a suitable assignment matrix even in a situation where the sizes of N^l and d^l are quite different.

Concatenation + Projection + Propagation: The above schemes consider both parts separately. To adaptively combine the two parts, this scheme further considers the node features after propagation $E^l[t]Z^l[t]$ as shown in Eqn. (7), which considers current node features and their neighbor nodes. Therefore, we propose the full scheme to generate the assignment matrix as Eqn. (5).

$$\begin{aligned} S^l_{CPP} &= softmax(\sigma([\sigma(E^l[t]W^l_E)||\mathcal{P}^l[t]]W^l_{CPP})),\\ \mathcal{P}^l[t] &= \sigma(E^l[t]Z^l[t]W^l_P)), \end{aligned} \tag{5}$$

where $W^l_E \in \mathbb{R}^{N^l\times H^l}$, $W^l_P \in \mathbb{R}^{d^l\times H^l}$, and $W^l_{CPP} \in \mathbb{R}^{2H^l\times N^{l+1}}$. In this way, we have the assignment matrix adaptively combining node features with correlations.

With the assignment matrix, we can learn hierarchical features $Z^l_h[t] = \{z^l_{m,h}[t]\}^{N^{l+1}}_{m=1}$ and hierarchical correlations $E^l_h[t] = \{e^l_{mn,h}[t]\}^{N^{l+1}}_{m,n=1}$ for the t-th graph in the layer l as Eqn. (6).

$$\begin{aligned} Z^l_h[t] &= S^l[t]^T Z^l[t] \in \mathbb{R}^{N^{l+1}\times d^l},\\ E^l_h[t] &= S^l[t]^T E^l[t] S^l[t] \in \mathbb{R}^{N^{l+1}\times N^{l+1}}. \end{aligned} \tag{6}$$

The sensors with similar properties are clustered by $S^l[t]$ and thus the local information between sensors is captured. Here, as our assignment matrix is designed specifically for MTS data, we are able to learn decent hierarchical features and hierarchical correlations for MTS data.

Then, we need to update the hierarchical features by capturing the hierarchical correlations, so that the spatial information can be utilized to learn better representations. We adopt Message Passing Neural Network (MPNN) [30], a kind of GNN, to achieve this. With MPNN, the information of each node is updated by the information propagated from its neighbour nodes, i.e., $Z^{l+1}[t] = \sigma(E^l_h[t]Z^l_h[t]W^l)$, where σ is the non-linear activation function (e.g., ReLU is adopted), and W^l is learnable parameters shared across sequential graphs to save training costs. The updated features $Z^{l+1}[t]$ are the input of the next layer. Meanwhile, the hierarchical correlations $E^{l+1}[t] = E^l_h[t]$ are used for the next layer. From node level, the propagation is divided into two processes, i.e., the propagation and updating processes, as shown in Fig. 6. In the propagation process, the information of neighbor nodes propagates to the central node, which then is updated via a neural network in the updating process. The general propagation process of each node can be represented as Eqn. (7), where $\mathcal{N}(m)$ is the neighbor set of node m. With MPNN, the features of each node are updated by the weighted average of neighbor nodes, so the hierarchical correlations are captured.

$$\begin{aligned} h^l_{m,h}[t] &= \sum_{j\in\mathcal{N}(m)} e^l_{mj,h}[t] z^l_{j,h}[t],\\ z^{l+1}_m[t] &= \sigma(h^l_{m,h}[t]W^l). \end{aligned} \tag{7}$$

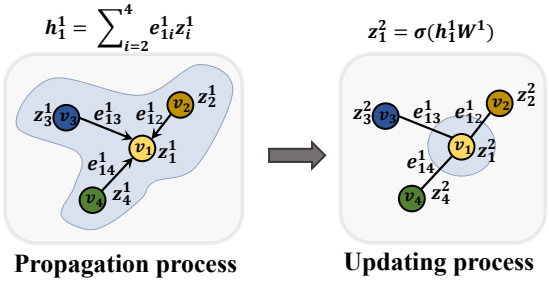

Fig. 6: The diagram of MPNN.

Algorithm 1 Framework of HierCorrPool

Input: An MTS sample $X = \{x_m\}_{m=1}^{N} \in \mathbb{R}^{N \times L}$, where each sensor m has signals with L timestamps, $x_m = \{x_m[1], x_m[2], ..., x_m[L]\}$;

Output: The learned decent representations $H \in \mathbb{R}^d$ for X;

1: Obtain feature windows $z_m = \{z_m[t]\}_{t=1}^{\bar{L}}$ for each sensor m from X.
2: Capture temporal dependencies and obtain learned feature windows $z'_m = \{z'_m[t]\}_{t=1}^{\bar{L}'}$ as Eqn. (1).
3: Construct sequential graphs $\mathcal{G} = \{G[t]\}_{t=1}^{\bar{L}'}$ based on the Eqn. (2).
4: **for** $i = 1$ to M **do** //M is the number of model layers
5: Learn $S^l[t] = \mathcal{S}(Z^l[t], E^l[t]; W^l)$ for each graph.
6: Learn $Z_h^l[t]$ and $E_h^l[t]$ as Eqn. (6).
7: Obtain $Z_h^{l+1}[t]$ by capturing $E_h^l[t]$ as Eqn. (7).
8: **end for**
9: Obtain H by stacking the hierarchical features.

After M layers of learning and capturing hierarchical correlations, the spatial dependencies in each graph are leveraged. Then, we map the learned hierarchical features in sequential graphs to learn overall representations. Suppose $D[t]$ as the node features after multiple layers to capture the hierarchical correlations, the final representations for a sample are derived as $H = \sigma([D[1]||D[2]||...||D[\bar{L}']]W)$. The pseudocode of our HierCorrPool can be shown in Algorithm 1. In this way, the dynamic property and hierarchical correlations are considered for representation learning. Therefore, the spatial-temporal information is better leveraged.

IV. Experimental Results

To evaluate the ability of our model on representation learning, we test our model on different tasks, including RUL prediction, SSC, and HAR. This section introduces the dataset details and shows the experimental results.

A. Dataset Descriptions and Experimental Settings

Three datasets are adopted for evaluation, including C-MAPSS for RUL prediction [31], UCI HAR dataset for HAR [32], and ISRUC-S3 for SSC [33], whose statistics are shown in TABLE I. These datasets consist of the signals collected from multiple sensors, and the signals show correlations between sensors.

C-MAPSS: The dataset describes the aircraft engine degradation, including four sub-datasets collected in different operating conditions and fault modes, where FD002 and FD004 have more conditions and modes than FD001 and FD003. To fully monitor the state of each engine, 21 sensors are deployed to measure different physical values. As the sensors with indices 1, 5, 6, 10, 16, 18, and 19 have constant values, we remove these sensors, and only 14 sensors keep for experiments [16]. As the datasets record the whole life cycle of engines, we here crop the signals to our desired dimension as traditional works did [16], [2]. Specifically, given an engine with the whole life cycle as $\mathcal{T}$, we adopt the time window with the fixed length L to slide along the signals by S steps for each sampling procedure. In this way, we can obtain the i-th MTS sample $X^i \in \mathbb{R}^{N \times L}$ and its RUL as $y^i = \mathcal{T} - L - i * S$. Meanwhile, we adopt the piece-wise linear RUL as the label of each sample [16], [2], i.e., we choose a predefined maximum RUL and set y^i as the value when y^i is larger than the predefined value. In particular, S and the maximum RUL are set as 1 and 125 respectively.

HAR dataset: The dataset describes 30 subjects performing six activities, including walking, walking upstairs, downstairs, standing, sitting, and lying down. Nine sensors are deployed to detect the activities. For training models, we split the data into 60%, 20%, and 20% for training, validation, and testing by following the preprocessing in [4].

ISRUC-S3: The dataset describes the sleep stages of ten subjects, and each recording contains ten channels. The sleep stages are divided into five stages, including wake, N1 stage, N2 stage, N3 stage, and REM, according to AASM standards [34]. For training models, we follow the setting in [18] and employ ten-fold cross-validation, i.e., we use the data from nine subjects for training and one subject for testing.

TABLE I: The statistic of datasets

Datasets		# Training	# Testing	# Sensors
C-MAPSS	FD001	13785	100	14
	FD002	30736	259	14
	FD003	15536	100	14
	FD004	33263	248	14
UCI HAR		7352	2947	9
ISRUC		7665	924	10

The experiments can be divided into three parts, including the comparisons with state-of-the-art models, the ablation study, and the sensitivity analysis. All methods were repeated 30 times, and the average results are shown to prevent randomness. Besides, we set batch size as 50, optimizer as Adam, learning rate as 0.001, and training epoch as 10 for training our model. Some hyperparameters of the correlation pooling layer will be discussed in the sensitivity analysis. Furthermore, we built our model based on Pytorch 1.9 and trained our model on NVIDIA GeForce RTX 3080Ti GPU.

To evaluate our model for RUL prediction, two metrics are adopted, including Root Mean Square Error (RMSE) and Score as used in [16], [2], [35]. Notably, Score gives more penalty for late predictions, which cause more serious

TABLE II: Comparisons with SOTA models on C-MAPSS for RUL prediction (RMSE)

Models	FD001		FD002		FD003		FD004	
	RMSE	P-Value	RMSE	P-Value	RMSE	P-Value	RMSE	P-Value
Hybrid [16]	14.53	4.08E-42	21.37	1.65E-56	13.24	1.27E-29	27.08	1.26E-60
MODBNE [36]	15.04	1.88E-44	25.05	7.89E-61	12.51	6.21E-24	28.66	4.54E-62
KDnet [2]	13.68	6.79E-37	14.47	3.70E-38	12.95	1.18E-27	15.96	2.59E-36
ATS2S [37]	12.63	2.69E-24	14.65	3.40E-39	11.44	9.83E-05	16.66	2.76E-40
CNN-LSTM [42]	12.47	2.16E-15	13.41	1.95E-19	12.02	5.53E-13	14.49	2.15E-05
AConvLSTM [43]	12.22	6.17E-03	12.50	1.97E-02	12.18	6.73E-18	**14.02**	5.49E-01
GCN [44]	12.87	7.63E-10	13.48	6.20E-11	12.93	3.95E-13	14.66	2.33E-09
DiffPool [21]	12.60	4.16E-12	13.06	9.17E-12	12.08	6.06E-10	15.06	1.52E-12
SAGPool [45]	12.66	5.93E-09	12.91	2.84E-08	12.38	1.52E-11	14.91	3.04E-12
iPool [46]	12.66	1.39E-07	13.40	1.27E-09	12.55	4.19E-12	14.85	1.25E-09
TAP [47]	12.81	7.76E-09	13.03	1.44E-12	12.81	1.12E-10	14.87	7.62E-11
Ours	**12.02**	-	**12.38**	-	**11.26**	-	14.08	-

consequences than early predictions. The lower RMSE and Score are, the better the model is. Besides, to evaluate our model on HAR and SSC, four metrics are adopted, including Accuracy (ACC), Macro-F1 score (MF1), Specificity (Spec.), and Sensitivity (Sens.). The higher the indicators are, the better the model is.

B. Comparison with State-of-the-art Models

This part compares our method with state-of-the-art (SOTA) methods. As our method is tested on different tasks, we compare with the SOTA methods specifically designed for the related areas for fair comparisons. For RUL prediction, we compare with Hybrid [16], MODBNE [36], KDnet [2], and AST2S [37], whose results are adopted from their source works. For HAR, we compare with SupEncoder [4], CPC [38], and TS-TCC [4], whose results are adopted from [4]. For SSC, we compare with DeepSleepNet [39], MVN [40], and SeqSleepNet [41], whose results are adopted from [18]. Besides, we also compare the methods designed for capturing spatial-temporal information within MTS data, including CNN-LSTM [42] and AConvLSTM [43], whose results are re-implemented on our datasets according to their setups. Further, we also compare with GNN-based methods, including the method which directly stacks sensor features, i.e., GCN [44], and the graph pooling methods specifically designed for graph data, including DiffPool [21], SAGPool [45], iPool [46], and TAP [47]. For fair comparisons, these GNN based methods use the same baseline as ours, i.e., the way for constructing sequential graphs. Besides, we also report the statistical significance test with P-Value for comparisons.

TABLE II and III show the results of RMSE and Score in C-MAPSS respectively. Compared with GCN which directly stacks sensor features, we observe from the results that most graph pooling methods can achieve better performance, indicating that it is useful to improve performance by capturing the hierarchical correlations between sensors. However, the current graph pooling methods cannot learn decent hierarchical correlations between sensors, thus restricting their performance. Compared with the second best graph pooling method, i.e., DiffPool, our method improves by 4.6%, 5.2%, 6.7%, and 6.5% respectively in terms of RMSE. As the GNN based methods have the same baseline with ours, the improvements indicate the effectiveness of our proposed correlation pooling scheme. Besides, compared with non-graph based methods, our method still achieves the best performance in most cases, such as FD001, FD002, and FD003, improving by 1.6%, 0.96%, and 1.5% respectively. In FD004, AConvLSTM achieves better performance than our method, i.e., 0.42%. This is because AConvLSTM also considers the spatial-temporal information within MTS data by stacking an MTS sample as a two-dimensional image and capturing sensor correlations by CNN, and meanwhile, they compute the attentions between adjacent timestamps, which gives them the powerful ability to process temporal information. However, the attention modules also bring them numerous operations and high computational costs, which is discussed in Section IV-E.

From the results in TABLE IV for HAR, we observe that our method still achieves better performance than SOTA methods. While most graph pooling methods achieve better performance than GCN and other traditional methods, our method can still achieve better performance than the graph pooling methods. Compared with the second best method, i.e., iPool, our method improves by 1.18%, 1.15%, 1.15%, and 0.23% respectively, indicating the effectiveness of our method. Meanwhile, the small P-Values can also prove the validity of the improvements. Similar improvements can also be observed in TABLE V for SSC. Compared with the second best results, i.e., TAP, our method improves by 1.06%, 1.21%, 1.3%, and 0.28%. These improvements indicate that our HierCorrPool can help learn better hierarchical correlations between sensors, and meanwhile, it is useful to capture the dynamic sensor correlations within MTS data. Thus, our method can achieve better performance than SOTA methods.

In addition, we use error bar charts to compare our method with some well-performing methods for evaluating the robustness. From Fig. 7 and 8, we observe that compared with SOTA methods, our method can achieve more stable performance. This is because our proposed correlation pooling scheme can generate the assignment matrix which considers the information from sensor features and correlations simultaneously, and thus we can learn better hierarchical correlations between sensors. Meanwhile, our sequential graphs can help us to capture the dynamic property within MTS data. Therefore, the

TABLE III: Comparisons with SOTA models on C-MAPSS for RUL prediction (Score)

Models	FD001		FD002		FD003		FD004	
	Score	P-Value	Score	P-Value	Score	P-Value	Score	P-Value
Hybrid [16]	322	5.74E-31	3077	3.43E-66	367	2.68E-38	5649	1.97E-65
MODBNE [36]	334	1.52E-32	5585	5.37E-75	421	1.25E-41	6557	1.25E-67
KDnet [2]	362	1.34E-35	929	8.25E-41	327	4.23E-35	1303	3.99E-35
ATS2S [37]	243	5.67E-07	876	1.86E-24	263	6.90E-29	1074	2.40E-16
CNN-LSTM [42]	257	2.99E-07	835	1.01E-12	252	6.58E-17	870	8.18E-01
AConvLSTM [43]	242	1.31E-02	661	5.42E-09	267	2.27E-16	886	5.26E-01
GCN [44]	315	5.19E-07	720	4.76E-06	300	9.07E-08	1125	9.08E-07
DiffPool [21]	298	1.21E-12	738	1.16E-08	247	1.86E-08	1275	.94E-09
SAGPool [45]	304	1.68E-10	727	2.66E-08	276	6.97E-09	1180	5.36E-10
iPool [46]	296	1.57E-07	790	1.18E-11	274	9.56E-11	1135	1.68E-09
TAP [47]	301	8.66E-11	707	1.64E-10	300	1.18E-08	1211	6.99E-09
Ours	**233**	-	**599**	-	**189**	-	**874**	-

TABLE IV: Comparisons with SOTA models on UCI HAR

Models	ACC		MF1		Sens.		Spec.	
	Avg.	P-Value	Avg.	P-Value	Avg.	P-Value	Avg.	P-Value
SupEncoder [4]	90.14	1.16E-33	90.31	2.48E-33	-	-	-	-
CPC [38]	83.85	4.36E-45	83.27	1.05E-45	-	-	-	-
TS-TCC [4]	92.93	2.03E-20	93.13	1.34E-19	-	-	-	-
CNN-LSTM [42]	82.86	9.86E-12	77.57	4.18E-12	81.18	4.12E-12	96.56	9.66E-12
AConvLSTM [43]	88.41	1.94E-09	88.32	4.07E-09	88.32	4.07E-09	97.68	1.68E-09
GCN [44]	92.62	6.51E-15	92.81	4.41E-15	92.77	4.02E-15	98.52	4.37E-15
DiffPool [21]	93.16	1.36E-09	93.33	4.21E-10	93.30	5.72E-10	98.63	1.02E-09
SAGPool [45]	93.12	4.05E-17	93.26	1.89E-18	93.27	3.87E-17	98.62	6.51E-17
iPool [46]	93.20	1.09E-08	93.36	2.25E-09	93.35	5.57E-09	98.64	1.15E-08
TAP [47]	93.09	7.61E-09	93.26	2.43E-09	93.24	3.84E-09	98.61	5.68E-09
Ours	**94.38**	-	**94.51**	-	**94.50**	-	**98.87**	-

TABLE V: Comparisons with SOTA models on ISRUC-S3 dataset for SSC

Models	ACC		MF1		Sens.		Spec.	
	Avg.	P-Value	Avg.	P-Value	Avg.	P-Value	Avg.	P-Value
DeepSleepNet [39]	78.80	1.41E-06	77.91	9.52E-05	-	-	-	-
MVN [40]	78.10	8.20E-08	76.80	3.7E-07	-	-	-	-
SeqSleepNet [41]	78.90	2.14E-06	76.30	3.35E-08	-	-	-	-
CNN-LSTM [42]	69.54	1.91E-10	58.33	7.45E-09	61.90	1.39E-09	91.88	2.11E-11
AConvLSTM [43]	71.95	4.53E-16	65.66	1.13E-17	65.08	2.14E-17	92.59	8.24E-16
GCN [44]	80.90	1.29E-03	78.41	7.29E-06	77.28	1.07E-04	95.00	2.17E-03
DiffPool [21]	80.91	1.91E-03	77.84	4.49E-06	76.40	1.06E-04	94.96	3.96E-03
SAGPool [45]	81.38	6.17E-04	78.70	5.04E-06	77.53	7.84E-05	95.09	1.16E-03
iPool [46]	81.49	3.13E-02	78.55	1.81E-04	77.45	2.53E-03	95.13	4.72E-02
TAP [47]	81.74	7.98E-02	79.19	1.98E-03	78.25	1.65E-02	95.20	1.08E-01
Ours	**82.80**	-	**80.40**	-	**79.55**	-	**95.48**	-

comprehensive information can lead to stable performance. Further, although AConvLSTM achieves better performance than ours in FD004, it has larger deviations. This is because AConvLSTM utilizes spatial information by a 2D CNN instead of GNN, while our method adopts GNN and learns hierarchical correlation between sensors at the same time, thus achieving stable performance.

C. Ablation Study

In this section, we conduct ablation study to test the effectiveness of our sequential graphs and the effect of different fusion schemes on model performance. First, we compare the model with sequential graphs to the one without sequential graphs (i.e., only one graph is constructed) to show the effectiveness of capturing dynamic sensor correlations. The first variant is GCN w/o Sequential graphs + direct stack, representing the model without sequential graphs and only constructing one graph for one sample as current works did [19], [20], and meanwhile, the sensor features are directly stacked to learn representations. The second variant is GCN with Sequential graphs + direct stack, where sequential graphs are used while the sensor features are still stacked directly instead of learning hierarchical correlations. At the same time, we evaluate the three variants of HierCorrPool, where concatenation, projection, and propagation are denoted as CAT, PROJ, and PROP respectively. The third variant is GCN + Sequential graphs + CAT, where we adopt our CAT fusion scheme to generate the assignment matrix for correlation pooling. The

TABLE VI: Ablation study for RUL prediction

Variants	FD001		FD002		FD003		FD004	
	RMSE	Score	RMSE	Score	RMSE	Score	RMSE	Score
GCN w/o Sequential graphs + direct stack	13.80	293	13.91	831	14.11	357	15.91	1266
GCN + Sequential graphs + direct stack	12.87	315	13.48	720	12.93	300	14.66	1125
GCN + Sequential graphs + CAT	12.28	252	12.49	625	11.61	208	14.39	934
GCN + Sequential graphs + CAT + PROJ	12.15	239	12.44	619	11.42	197	14.19	908
GCN + Sequential graphs + CAT + PROJ + PROP	12.02	233	12.38	599	11.26	189	14.08	874

TABLE VII: Ablation study for HAR and SSC

Variants	HAR				ISRUC-S3			
	ACC	MF1	Sens.	Spec.	ACC	MF1	Sens.	Spec.
GCN w/o Sequential graphs + direct stack	91.83	91.83	91.79	98.36	79.67	75.31	75.36	94.53
GCN + Sequential graphs + direct stack	92.62	92.81	92.77	98.52	80.90	78.41	77.28	95.00
GCN + Sequential graphs + CAT	93.98	94.10	94.11	98.79	82.07	79.20	78.17	95.29
GCN + Sequential graphs + CAT + PROJ	94.16	94.28	94.29	98.83	82.54	80.12	79.21	95.41
GCN + Sequential graphs + CAT + PROJ + PROP	94.38	94.51	94.50	98.87	82.80	80.40	79.55	95.48

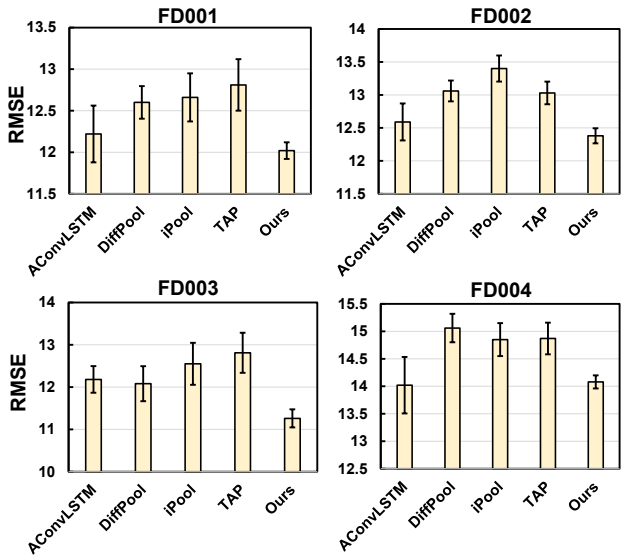

Fig. 7: The diagram of the error bars for RUL in terms of RMSE.

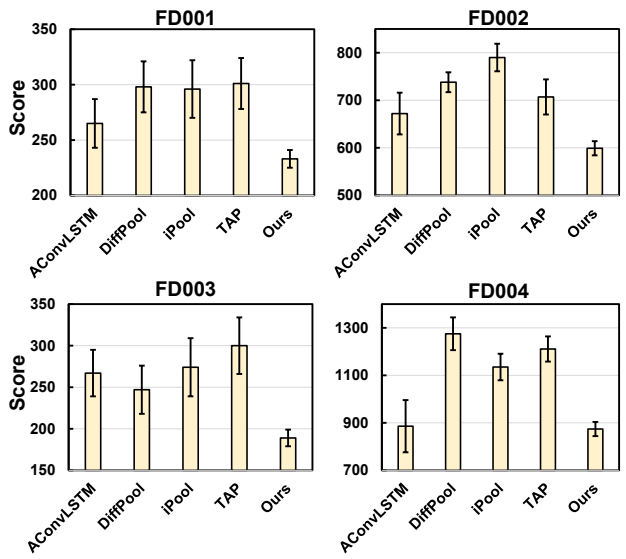

Fig. 8: The diagram of the error bars for RUL in terms of Score.

fourth one is GCN + Sequential graphs + CAT + PROJ, where the projection matrix is introduced. The final one is our complete version. TABLE VI and VII show the results for different tasks.

From the results, we first observe that the sequential graphs can help to improve the model performance due to considering the dynamic sensor correlations within MTS data. Let's take the results of RMSE in FD001 as an example. The model with sequential graphs improves by 6.7% compared to the one without sequential graphs. However, the direct stacking for sensor features still hinders performance due to the ignorance of hierarchical correlations between sensors. From the results, we observe that even CAT can achieve better performance than the model directly stacking sensor features, improving by 4.6% in FD001. Besides, from the results between CAT and CAT + PROJ, we observe that CAT + PROJ has equal performance to CAT in FD002 and better performance in the other datasets. The improvements indicate that CAT + PROJ can be applicable to more situations due to the added projection functions. Besides, we can also find that CAT + PROJ + PROP has better performance than CAT + PROJ in most cases.

These results show that the learned sequential graphs can be useful to improve performance due to dynamic sensor correlations captured, and that our proposed correlation pooling scheme can further improve performance and also make performance consistently good in most cases.

D. Sensitivity Analysis

As our method builds multiple correlation pooling layers to learn hierarchical sensor correlations, it is necessary to analyze the effect of the number of layers on final performance.

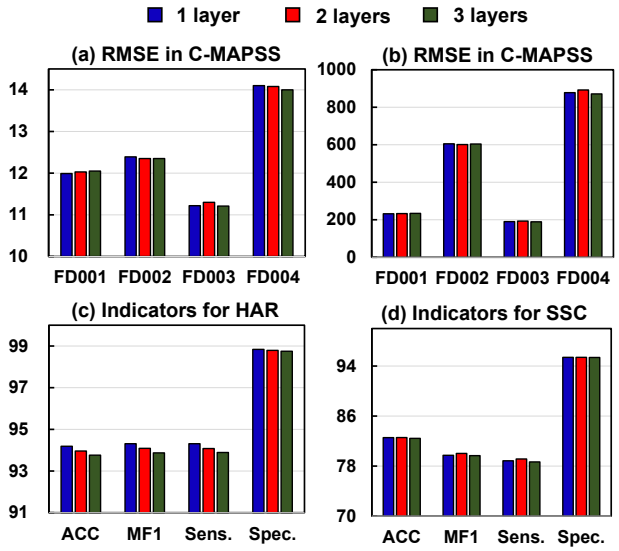

Fig. 9: The effect of the number of correlation pooling layers.

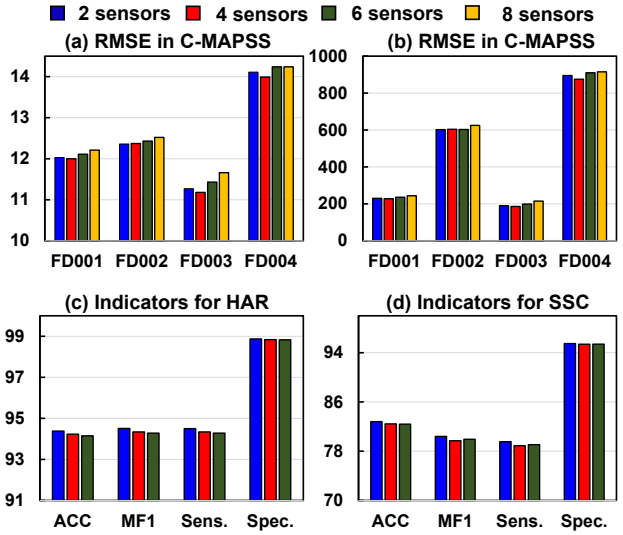

Fig. 10: The effect of the number of sensors in the last layer.

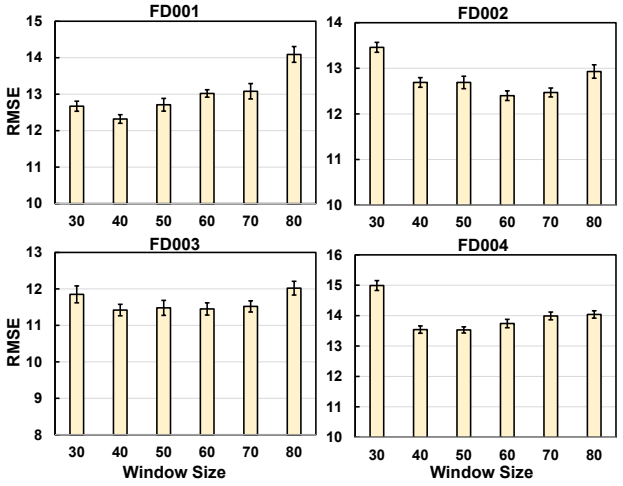

Fig. 11: The analysis for window size (RMSE).

Besides, the sensor correlations become coarse (i.e., many sensors are clustered as few sensors) after correlation pooling, and the sensors in the last layer affect final performance. Therefore, it is necessary to explore how many sensors in the last layer are best for the final performance. Further, for each MTS sample $X \in \mathbb{R}^{L \times N}$, the size of time window L may affect the final performance, so we also evaluate the effect of window size to provide the guideline on how to select the optimum size of the time window.

1) Number of correlation pooling layers:

Fig. 9 shows the effect of the number of layers on final performance. From the results, two points can be derived. First, the increase of correlation pooling layers can improve the performance of representation learning for some tasks. For example, the model with two layers shows improvements in FD002 and FD004 compared to the model with one layer, indicating the positive effect of increasing correlation pooling layers. Second, too many layers, however, are not always better and even hurt the final performance, such as in UCI-HAR. This is caused by over-smoothing in GNN [48], i.e., more GNN layers will make the information between neighbor nodes over-smooth in the message passing process, leading to features between nodes converging to the same values. Therefore, the suitable number of layers needs to be determined in the design of our model. From the results, it can be found that a few layers (i.e., 1 or 2 layers) would be good enough for learning representations of MTS data.

2) Number of sensors in the last layer:

We choose at most eight sensors in the last layer for C-MAPSS since this dataset consists of 14 sensors in total and six sensors at most for UCI HAR and ISRUC since these datasets consist of only nine and ten sensors respectively. Fig. 10 shows the effect of the number of sensors on final performance. From the results, it can be found that too few sensors in the last layer may cause poor performance. For example, our model achieves the worst performance on FD001, FD003, and FD004 for RUL prediction and UCI HAR for HAR when only two sensors are set in the last layer. This is because too few sensors in the last layer cannot fully reflect the correlations between sensors. It is noted that our model utilizes the proposed correlation pooling scheme to cluster together the sensors with high correlations. However, uncorrelated sensors may be clustered together when a small number is set in the last layer, which may ruin the learned hierarchical correlations. Therefore, it is required to choose a suitable number of sensors in the last layer. From Fig. 10, we can find that four sensors for C-MAPSS, two sensors for UCI HAR and ISRUC are better than the other choices. With the consideration that there are 14 sensors in C-MAPSS, 9 sensors in UCI HAR, 10 sensors in ISRUC, we can derive that our model achieves better performance when around one-third sensors are clustered.

3) Window size analysis: As the window sizes of UCI-HAR and ISRUC-S3 are fixed, we only analyze the effect of window sizes in C-MAPSS. Fig. 11 and 12 show the analysis in terms of RMSE and Score respectively, where we adopt the size ranging from 30 to 80 with an interval of 10. Taking the

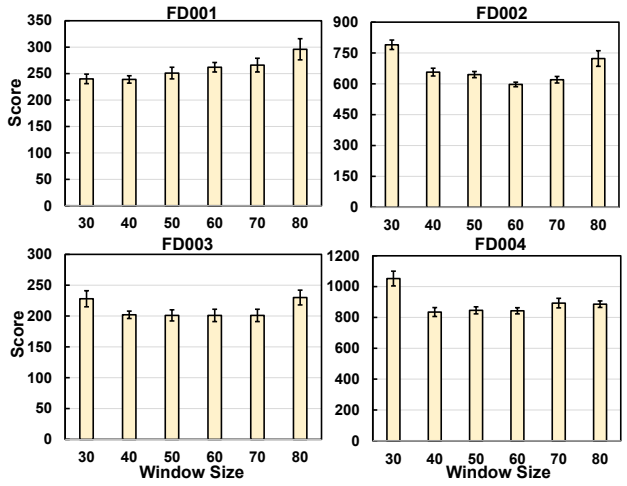

Fig. 12: The analysis for window size (Score)

results of RMSE in Fig. 11 as an example, we observe that our model generally achieves good performance in FD001 and FD003 when the window size is set as a small value, e.g., 40. But a too small window cannot bring better performance; on the contrary, the performance may become worse, e.g., our model achieves worse performance when the window size is 30 in FD003. Meanwhile, we can also observe that our model generally achieves better performance in FD002 and FD004 when the window size is set as a large value, e.g., 60. But the performance becomes worse when the value is larger than 60. It is noted that FD002 and FD004 contain more operating conditions and fault modes, so they can be seen as more complicated than FD001 and FD003. Therefore, we can derive from the results that large windows for complicated datasets and small windows for simple datasets are good for our model to achieve better performance. However, the sizes cannot be too large or too small.

E. Model Complexity

Model complexity is a key factor in determining whether a model is applicable to the real world. A model will be meaningless if it is too complex to be deployed in real systems, even though it is able to achieve great performance. Therefore, we test the model complexity in this section. We here evaluate the computational and storage complexity by using FLoating-point Operations Per second (FLOPs) and the number of model weights respectively. Meanwhile, the training and inference time are also compared. Notably, we adopt the running time in one epoch for the training time, as all methods run the same number of epochs. Further, we simulate the process in real systems to record the inference time, i.e., we predict the samples in a testing dataset one by one and calculate the average time for predicting one sample.

We choose the four most competitive methods based on the results in TABLE II, III, IV, and V, i.e., AConvLSTM, DiffPool, iPool, and TAP, to compare with our method for model complexity. TABLE VIII shows the comparisons. From the results, we observe that our method requires a little more FLOPs and model weights than DiffPool. This is because our method needs extra modules to learn decent assignment matrices for MTS data, compared with DiffPool. However, the increased model complexity is very marginal, e.g., increasing by 0.04% and 0.01% regarding FLOPs and model weights respectively, while our method has remarkable improvements compared to DiffPool, e.g., 4.6% in FD001 of C-MAPSS. Further, we also observe that AConvLSTM requires fewer model weights than ours; however, they need a large number of FLOPs, which also incurs much more training and inference time. From the results, we can derive that our method requires reasonable computation and storage costs and can be thus applicable to real systems.

TABLE VIII: The comparisons with SOTAs for model complexity

Models	FLOPs	# Weights	Training/s	Inference/ms
AConvLSTM	17,682,128	**67,943**	18.61	27.75
DiffPool	**939,441**	171,261	**1.16**	**1.19**
iPool	1,537,480	191,085	1.57	2.83
TAP	1,356,904	300,587	1.62	2.62
Ours	939,889	171,295	1.19	1.21

V. Conclusion

To better leverage the spatial-temporal information within MTS data, we propose a Hierarchical Correlation Pooling boosted graph neural network (HierCorrPool). We first learn sequential graphs to represent the dynamic sensor correlations within MTS data. Second, we learn better hierarchical correlations between sensors by proposing a novel correlation pooling scheme. Specifically, a new assignment matrix is designed for MTS data by considering both sensor features and correlations. Our model is tested on different tasks for evaluating the performance of representation learning, and the results show that capturing dynamic hierarchical correlations between sensors can bring better performance for our method, and meanwhile, our proposed correlation pooling scheme is suitable for learning hierarchical correlations from MTS data.

As the training samples in real systems may be difficult to obtain, we will focus on combining our proposed method with few-shot learning techniques [49] to save costs for collecting data in our future work.

References

[1] B.-L. Lu, Z.-H. Liu, H.-L. Wei, L. Chen, H. Zhang, and X.-H. Li, "A deep adversarial learning prognostics model for remaining useful life prediction of rolling bearing," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 4, pp. 329–340, 2021.

[2] Q. Xu, Z. Chen, K. Wu, C. Wang, M. Wu, and X. Li, "Kdnet-rul: A knowledge distillation framework to compress deep neural networks for machine remaining useful life prediction," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 2, pp. 2022–2032, 2022.

[3] T. Ahmad, L. Jin, X. Zhang, S. Lai, G. Tang, and L. Lin, "Graph convolutional neural network for human action recognition: A comprehensive survey," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 2, pp. 128–145, 2021.

[4] E. Eldele, M. Ragab, Z. Chen, M. Wu, C. K. Kwoh, X. Li, and C. Guan, "Time-series representation learning via temporal and contextual contrasting," in *IJCAI*, 2021, pp. 2352–2359.

[5] Y. Lei, N. Li, S. Gontarz, J. Lin, S. Radkowski, and J. Dybala, "A model-based method for remaining useful life prediction of machinery," *IEEE Transactions on Reliability*, vol. 65, no. 3, pp. 1314–1326, 2016.

[6] G. E. P. Box and D. A. Pierce, "Distribution of residual autocorrelations in autoregressive-integrated moving average time series models," *Journal of the American Statistical Association*, vol. 65, no. 332, pp. 1509–1526, 1970.

[7] S. Seto, W. Zhang, and Y. Zhou, "Multivariate time series classification using dynamic time warping template selection for human activity recognition," in *2015 IEEE Symposium Series on Computational Intelligence*, 2015, pp. 1399–1406.

[8] R. Mousheimish, Y. Taher, and K. Zeitouni, "Automatic learning of predictive cep rules: Bridging the gap between data mining and complex event processing," in *Proceedings of the 11th ACM International Conference on Distributed and Event-Based Systems*. New York, NY, USA: Association for Computing Machinery, 2017, p. 158–169.

[9] M. G. Baydogan and G. Runger, "Learning a symbolic representation for multivariate time series classification," *Data Mining and Knowledge Discovery*, vol. 29, p. 400–422, 2015.

[10] A. Gupta, H. P. Gupta, B. Biswas, and T. Dutta, "Approaches and applications of early classification of time series: A review," *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 1, pp. 47–61, 2020.

[11] G. Dewangan and S. Maurya, "Fault diagnosis of machines using deep convolutional beta-variational autoencoder," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 2, pp. 287–296, 2022.

[12] J. Zhu, N. Chen, and W. Peng, "Estimation of bearing remaining useful life based on multiscale convolutional neural network," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 4, pp. 3208–3216, 2019.

[13] l. Wen, Y. Dong, and L. Gao, "A new ensemble residual convolutional neural network for remaining useful life estimation," *Mathematical biosciences and engineering*, vol. 16, no. 2, p. 862–880, 2019.

[14] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, "Long short-term memory network for remaining useful life estimation," in *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, 2017, pp. 88–95.

[15] R. N. Khushaba, A. Phinyomark, A. H. Al-Timemy, and E. Scheme, "Recursive multi-signal temporal fusions with attention mechanism improves emg feature extraction," *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 2, pp. 139–150, 2020.

[16] Z. Chen, M. Wu, R. Zhao, F. Guretno, R. Yan, and X. Li, "Machine remaining useful life prediction via an attention-based deep learning approach," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 3, pp. 2521–2531, 2021.

[17] T. Li, Z. Zhou, S. Li, C. Sun, R. Yan, and X. Chen, "The emerging graph neural networks for intelligent fault diagnostics and prognostics: A guideline and a benchmark study," *Mechanical Systems and Signal Processing*, vol. 168, p. 108653, 2022.

[18] Z. Jia, Y. Lin, J. Wang, X. Ning, Y. He, R. Zhou, Y. Zhou, and L.-w. H. Lehman, "Multi-view spatial-temporal graph convolutional networks with domain generalization for sleep stage classification," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 29, pp. 1977–1986, 2021.

[19] Y. Zhang, Y. Li, Y. Wang, Y. Yang, and X. Wei, "Adaptive spatio-temporal graph information fusion for remaining useful life prediction," *IEEE Sensors Journal*, vol. 22, no. 4, pp. 3334–3347, 2021.

[20] Z. Jia, Y. Lin, J. Wang, X. Ning, Y. He, R. Zhou, Y. Zhou, and H. L. Li-wei, "Multi-view spatial-temporal graph convolutional networks with domain generalization for sleep stage classification," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 29, pp. 1977–1986, 2021.

[21] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *Advances in neural information processing systems*, vol. 31, 2018.

[22] C.-L. Liu, W.-H. Hsaio, and Y.-C. Tu, "Time series classification with multivariate convolutional neural network," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 6, pp. 4788–4797, 2019.

[23] K. Wang, K. Li, L. Zhou, Y. Hu, Z. Cheng, J. Liu, and C. Chen, "Multiple convolutional neural networks for multivariate time series prediction," *Neurocomputing*, vol. 360, pp. 107–119, 2019.

[24] C.-L. Yang, C.-Y. Yang, Z.-X. Chen, and N.-W. Lo, "Multivariate time series data transformation for convolutional neural network," in *IEEE International Symposium on System Integration*, 2019, pp. 188–192.

[25] S. Du, T. Li, Y. Yang, and S.-J. Horng, "Multivariate time series forecasting via attention-based encoder–decoder framework," *Neurocomputing*, vol. 388, pp. 269–279, 2020.

[26] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4027–4035.

[27] Z. Jia, Y. Lin, J. Wang, R. Zhou, X. Ning, Y. He, and Y. Zhao, "Graphsleepnet: Adaptive spatial-temporal graph convolutional networks for sleep stage classification." in *IJCAI*, 2020, pp. 1324–1330.

[28] T. Li, Z. Zhao, C. Sun, R. Yan, and X. Chen, "Multireceptive field graph convolutional networks for machine fault diagnosis," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 12, pp. 12 739–12 749, 2021.

[29] R. Jin, M. Wu, K. Wu, K. Gao, Z. Chen, and X. Li, "Position encoding based convolutional neural networks for machine remaining useful life prediction," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 8, pp. 1427–1439, 2022.

[30] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," *arXiv preprint arXiv:1704.01212*, 2017.

[31] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *International Conference on Prognostics and Health Management*, 2008, pp. 1–9.

[32] D. Anguita, A. Ghio, L. Oneto, F. Parra, and J. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 01 2013.

[33] S. Khalighi, T. Sousa, J. M. Santos, and U. Nunes, "Isruc-sleep: A comprehensive public dataset for sleep researchers," *Computer Methods and Programs in Biomedicine*, vol. 124, pp. 180–192, 2016.

[34] R. Berry, R. Budhiraja, D. Gottlieb, D. Gozal, C. Iber, V. Kapur, C. Marcus, R. Mehra, S. Parthasarathy, S. Quan, S. Redline, K. Strohl, S. Ward, and M. Tangredi, "Rules for scoring respiratory events in sleep: Update of the 2007 aasm manual for the scoring of sleep and associated events," *Journal of clinical sleep medicine*, vol. 8, pp. 597–619, 10 2012.

[35] C.-G. Huang, H.-Z. Huang, and Y.-F. Li, "A bidirectional lstm prognostics method under multiple operational conditions," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 11, pp. 8792–8802, 2019.

[36] "Generative adversarial networks based remaining useful life estimation for iiot," *Computers Electrical Engineering*, vol. 92, p. 107195, 2021.

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

[38] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[39] A. Supratak, H. Dong, C. Wu, and Y. Guo, "Deepsleepnet: A model for automatic sleep stage scoring based on raw single-channel eeg," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 11, pp. 1998–2008, 2017.

[40] S. Chambon, M. N. Galtier, P. J. Arnal, G. Wainrib, and A. Gramfort, "A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 26, no. 4, pp. 758–769, 2018.

[41] H. Phan, F. Andreotti, N. Cooray, O. Y. Chén, and M. De Vos, "Seqsleepnet: End-to-end hierarchical recurrent neural network for sequence-to-sequence automatic sleep staging," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, no. 3, pp. 400–410, 2019.

[42] O. Nafea, W. Abdul, G. Muhammad, and M. Alsulaiman, "Sensor-based human activity recognition with spatio-temporal deep learning," *Sensors*, vol. 21, no. 6, p. 2141, 2021.

[43] Y. Xiao, H. Yin, Y. Zhang, H. Qi, Y. Zhang, and Z. Liu, "A dual-stage attention-based conv-lstm network for spatio-temporal correlation and multivariate time series prediction," *International Journal of Intelligent Systems*, vol. 36, no. 5, pp. 2036–2057, 2021.

[44] M. Welling and T. N. Kipf, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2016.

[45] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *International conference on machine learning*. PMLR, 2019, pp. 3734–3743.

[46] X. Gao, W. Dai, C. Li, H. Xiong, and P. Frossard, "ipool–information-based pooling in hierarchical graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[47] H. Gao, Y. Liu, and S. Ji, "Topology-aware graph pooling networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 12, pp. 4512–4518, 2021.

[48] G. Li, M. Müller, A. Thabet, and B. Ghanem, "Deepgcns: Can gcns go as deep as cnns?" in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9266–9275.

[49] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.

Yucheng Wang received the B.Eng. degree from Central South University in 2018 and the M.Eng from Huazhong University of Science and Technology, China in 2021. He is a research engineer at Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore, and a PhD student at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests include deep learning, graph neural network, time series and related applications.

Min Wu is currently a senior scientist in Machine Intellection Department, Institute for Infocomm Research (I2R), Agency for Science, Technology and Research (A*STAR), Singapore. He received his Ph.D. degree in Computer Science from Nanyang Technological University (NTU), Singapore, in 2011 and B.E. degree in Computer Science from University of Science and Technology of China (USTC) in 2006. He received the best paper awards in IEEE ICIEA 2022, IEEE SmartCity 2022, InCoB 2016 and DASFAA 2015, and the finalist academic paper award in IEEE PHM 2020. He also won the CVPR UG2+ challenge in 2021 and the IJCAI competition on repeated buyers prediction in 2015. His current research interests include machine learning, data mining and bioinformatics.

Xiaoli Li 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). Xiaoli has published more than 260 high quality papers and won 8 best paper/benchmark competition awards.

Lihua Xie received the Ph.D. degree in electrical engineering from the University of Newcastle, Australia, in 1992. Since 1992, he has been with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, where he is currently a professor and Director, Center for Advanced Robotics Technology Innovation. He served as the Head of Division of Control and Instrumentation and Co-Director, Delta-NTU Corporate Lab for Cyber-Physical Systems. He held teaching appointments in the Department of Automatic Control, Nanjing University of Science and Technology from 1986 to 1989. Dr Xie's research interests include robust control and estimation, networked control systems, multi-agent networks, smart sensing and unmanned systems. He is an Editor-in-Chief for Unmanned Systems and has served as Editor of IET Book Series in Control and Associate Editor of a number of journals including IEEE Transactions on Automatic Control, Automatica, IEEE Transactions on Control Systems Technology, IEEE Transactions on Network Control Systems, and IEEE Transactions on Circuits and Systems-II. He was an IEEE Distinguished Lecturer (Jan 2012 – Dec 2014). Dr Xie is Fellow of Academy of Engineering Singapore, IEEE, IFAC, and CAA.

Zhenghua Chen received the B.Eng. degree in mechatronics engineering from University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2011, and Ph.D. degree in electrical and electronic engineering from Nanyang Technological University (NTU), Singapore, in 2017. Currently, he is a Scientist and Lab Head at Institute for Infocomm Research, and an Early Career Investigator at Centre for Frontier AI Research (CFAR), 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, Best Paper Award at IEEE ICIEA 2022 and IEEE SmartCity 2022, etc. He serves as Associate Editor for Elsevier Neurocomputing and IEEE Transactions on Instrumentation and Measurement. He is currently the Chair of IEEE Sensors Council Singapore Chapter and IEEE Senior Member. His research interests include data-efficient and model-efficient learning with related applications in smart city and smart manufacturing.