International Journal of Innovative
Computing, Information and Control
Volume **18**, Number **5**, October **2022**

ICIC International ©2022 ISSN 1349-4198
pp. **1363–1381**

MULTI-SENSOR DATA FUSION BASED ON GCN-LSTM

Bohuai Xiao[2], Xiaolan Xie[1,2,*] and Chengyong Yang[3,*]

[1]Guangxi Key Laboratory of Embedded Technology and Intelligent System
[2]School of Information Science and Engineering
[3]Network and Information Center
Guilin University of Technology
No. 319, Yanshan Street, Yanshan District, Guilin 541004, P. R. China
xbh@glut.edu.cn; *Corresponding authors: { xxl; ychy }@glut.edu.cn

Received February 2022; revised May 2022

Abstract. *Due to the constraints of traditional multi-sensor systems with multiple sources and heterogeneity, the common fusion model is inefficient and error-prone, which is difficult to meet the demand for multi-sensor data fusion. To cope with this problem, this paper proposes a combined graph convolutional network (GCN) and long short-term memory (LSTM) algorithm to construct a feature-level fusion model for sensor data. The model uses GCN to extract features of multi-source heterogeneous data, which solves the problem of difficult fusion of heterogeneous data caused by differences in data types, and LSTM for feature extraction of time series, which solves the problem of gradient disappearance. Finally, the proposed model is validated on an industrial-grade data set, and the test results show the model's effectiveness.*

Keywords: Multi-sensor data fusion, Graph convolutional network, Long short-term memory

1. **Introduction.** With the continuous progress in science and technology, our intelligent world is surrounded by enormous amounts of sensor data that are highly desirable for efficient caching, computing, networking, and security [1]. Although the amount of data from various sensors is growing exponentially, not all data between sensors can be interpreted or described uniformly [2]. Therefore, multi-sensor data fusion technology is an emerging research area on data processing for the specific problem of using multiple sensors in a system [3]. The basic principle of multi-sensor data fusion technology is like the integrated processing of information in the human brain, making full use of multiple sensor resources, through the rational domination and use of multiple sensors and their observation information, combining redundant or complementary information from multiple sensors in space or time according to some criteria, in order to obtain a consistent interpretation or description of the object under test [4,5].

Although a complete theoretical system and efficient fusion algorithms have not yet been developed for multi-sensor data fusion, many mature and effective fusion models have been proposed in many application areas according to their specific application contexts, especially those combined with artificial intelligence algorithms. Gao et al. improved back propagation (BP) neural network by particle swarm optimization (PSO) and improved the accuracy of sensor data fusion [6]. Jiao et al. combined the Kalman filter-support vector machine (KF-SVM) improved the accuracy of sensor data fusion and obtained better state prediction [7]. Qi et al. used a long short-term memory (LSTM) module and a multilayer recurrent neural network (RNN) to form a multi-sensor data fusion model that can achieve

DOI: 10.24507/ijicic.18.05.1363

a higher recognition rate and faster inference speed [8]. Lima et al. stated that nonlinear autoregressive networks with exogenous input (NARX) and multiple linear regression (MLR) models were utilized to improve the accuracy of multi-sensor data fusion mass flow prediction [9]. It can be expected that new concepts and techniques such as neural networks and artificial intelligence will play an increasingly important role in multi-sensor data fusion. However, the challenging problem of multi-sensor data fusion algorithms is still far from complete. Taking the above paper as an example, most artificial neural network models can only process conventional Euclidean structured data, and tend to ignore the multi-source and temporal nature of the data during multi-sensor data fusion, which leads to deviation of feature extraction results from the actual situation.

Recent years have seen numerous attempts in deep learning in artificial intelligence and machine learning to vitalize the graph neural network (GNN) [10]. More and more practical processing tasks in real life can be abstracted into graph data structures, and the data objects processed by GNN are non-Euclidean data structures with irregular structures. Among them, graph convolutional network (GCN) [11] is an essential branch of GNN with high neural network interpretation, which is often used to extract non-Euclidean spatial features. The long short-term memory (LSTM) network is a variant of recurrent neural network (RNN) [12], which is often used to extract temporal features and solve the gradient disappearance and gradient explosion problems during the training of long sequences. In models based on non-Euclidean spatial and temporal feature extraction, the combination of GCN and LSTM can effectively improve the performance of data fusion models in space and time. In a multi-sensor system, the environmental information provided by each information source has differences, and the fusion process of these disparate information is actually a heterogeneous inference process [13].

This paper proposes an algorithm on GCN combined with LSTM, which takes multi-source sensor data as input, extracts spatial features of multi-source sensor data using GCN, performs feature extraction of time series by LSTM, and then achieves feature-level fusion of multi-sensor data.

Section 2 discusses related work on data fusion models, such as the basic principles of GCN and LSTM. Then, we propose the algorithm for combining GCN and LSTM in Section 3, and simulation experiments are performed in Section 4. Finally, Section 5 summarizes the contributions of this study.

2. **Related Work.**

2.1. **Multi-sensor data fusion based on artificial neural network.**

Neural networks are highly fault-tolerant, self-learning, self-organizing and self-adaptive, capable of simulating complex nonlinear mappings. These characteristics of neural networks and their powerful nonlinear processing capabilities precisely meet the requirements of multi-sensor data fusion technology processing. In a multi-sensor system, the environmental information provided by each information source has a certain degree of uncertainty, and the fusion process of this uncertain information is actually an uncertainty inference process [14,15]. Neural networks determine the classification criteria based on the similarity of the samples accepted by the current system, and this determination method is mainly expressed in the distribution of the weights of the network. At the same time, a learning algorithm combining artificial neural networks and recurrent architecture can be used to acquire knowledge and obtain uncertainty inference mechanisms [16]. Using the signal processing capability and automatic inference function of neural networks, as shown in Figure 1, multi-sensor data fusion is achieved.

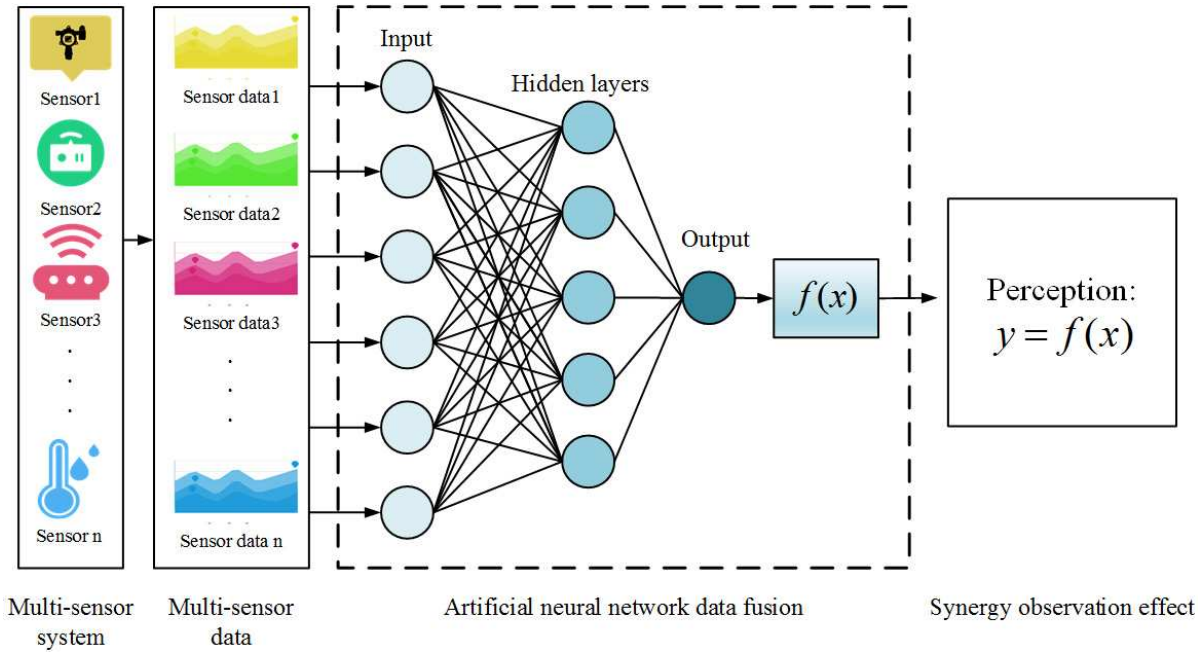


FIGURE 1. A framework for multi-sensor system fusion based on artificial neural networks

2.2. **GCN.** GCN is a kind of convolutional neural network which is studied with graph data, and its emergence provides new ideas for processing some non-Euclidean graph data [17,18], for example, GCN can be applied to social network analysis, recommendation systems, traffic flow prediction, etc. In a multi-sensor environment, the sensor data can be regarded as non-Euclidean graph data, and GCN has good feature extraction capability for non-Euclidean data structures. Therefore, we adopt GCN to handle spatial feature extraction and mapping of sensor data.

The essential purpose of GCN is to use graph convolution to extract spatial features of non-Euclidean structured graph data, and for graphs $G = (V, E, A)$, the input signals X and Y, the processing taken by GCN is shown in Equation (1).

$$f(X, A) = Y \tag{1}$$

where $V = \{v_i\}_{i=1}^{N}$ denotes the number of nodes in the graph; $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ denotes the adjacency matrix of the graph; and the elements A_{ij} in the matrix A denote the connection between the nodes and the graph.

The forward propagation formula for the graph convolution is shown in Equation (2).

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{l}W^{l}\right) \tag{2}$$

where $\tilde{A} = A + I$, I is a matrix of size $N \times N$; $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ denotes the diagonal matrix; $H^l \in R^{N \times D}$ denotes the output value of l layer, where $H^0 = X$; $\sigma(\cdot)$ denotes the activation function; W^l denotes the parameter value of l layer.

2.3. **LSTM.** LSTM networks were first proposed by Hochreiter and Schmidhuber in 1997 to solve the long-term dependency problem in RNNs [19]. As shown in Figure 2(a), although each RNN unit will pass the information extracted by itself to the next unit, information loss will occur when the length of the passing chain increases to a certain degree, which is the problem of long-term dependence mentioned above, and this prompted

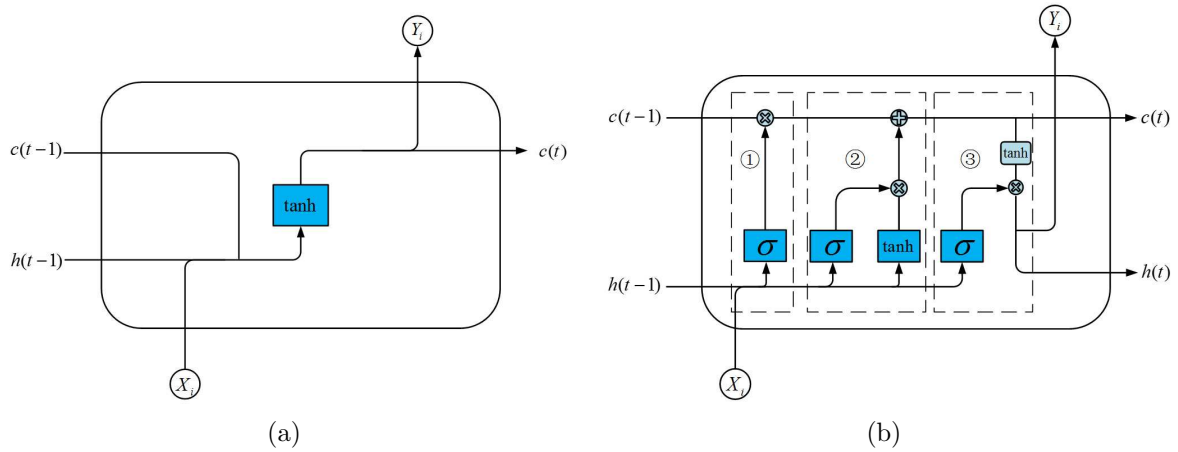


FIGURE 2. (a) Traditional RNN structure; (b) RNN's variant LSTM structure, where the first part is the forgetting gate f, the second department is the input gate i, and the third part is the output gate o

the emergence and popularity of LSTM. RNN has no cell state, while LSTM can remember information through cell state; RNN activation function can only use tanh function, while LSTM introduces sigmoid function combined with tanh function, adds summation operation, and reduces the possibility of gradient disappearance and gradient explosion; RNN can only deal with short-term dependence problems, while LSTM can deal with both short-term dependence problem and LSTM can deal with both short-term and long-term dependence problems. Therefore, we adopt LSTM to handle temporal feature extraction and mapping of sensor data. Specifically, as shown in Figure 2(b), the LSTM uses three types of gating units, namely forgetting gates f, input gates i and output gates o [20].

The forgetting gate f is used to control what information is to be discarded from the current cell state C_t. First, the hidden state information at moment $t-1$ is input to the sigmoid function together with the data at moment t. The output value is on the interval $[0, 1]$, where larger means more remembered and smaller means more should be forgotten. Finally, the obtained value is multiplied with the cellular memory C_{t-1} of the previous moment as a way to limit the influence of the previous memory on the later. The state update equation of the forgotten gate f is shown in Equation (3).

$$f_t = \sigma\left(W_f\left[h_{t-1}, x_t\right] + b_f\right) \tag{3}$$

The input gate i controls how much of the current network input x_t will be retained in the cell state C_t. First, the memory C_{t-1} at moment $t-1$ is operated by the same memory measurement rule as the forgetting gate f, which serves to truncate and filter past memories. Then, the result is added to C_{t-1} by doing the operation with the combined input adjusted by the tanh function. Finally, the total capacity of the memory is extended so that the memory can be constantly updated. The rules for updating the state of the input gates are shown in Equations (4)-(6).

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{4}$$

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{5}$$

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{6}$$

Output gate o controls how much information can be output for the current cell state C_t. First, the information passed by the input gate is adjusted with the tanh function. Then, the hidden state at the previous moment $t-1$ is multiplied with the integrated

data input at this moment t. The product operation is done. Finally, the computed data is output and transferred to the next LSTM unit. The state update rules for the output gate are shown in Equations (7) and (8).

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{7}$$

$$h_t = o_t \times \tanh(C_t) \tag{8}$$

where σ denotes the activation function that generates a vector between $[0, 1]$ based on the input; $\tilde{C}_t$ denotes the candidate cell information; W_f, W_i, W_o, W_c denote the weight coefficient matrix in the LSTM cell state update process; b_f, b_i, b_o, b_c denote the bias matrix in the state update process.

3. A GCN-LSTM Model for Multi-Sensor Data Fusion.

3.1. Basic ideas. The multi-sensor data fusion process is abstracted as a graph model, as shown in Figure 3, in which, the individual sensor information is abstracted as nodes and the connection information between individual sensors is abstracted as the edges of the graph model. The features within each sensor are used as the feature vectors of each node, and the sensor connectivity information is embedded in the connectivity relationships of the graph structure. At the same time, the states of nodes in the graph model depend on their own states and the influence of other states [21].

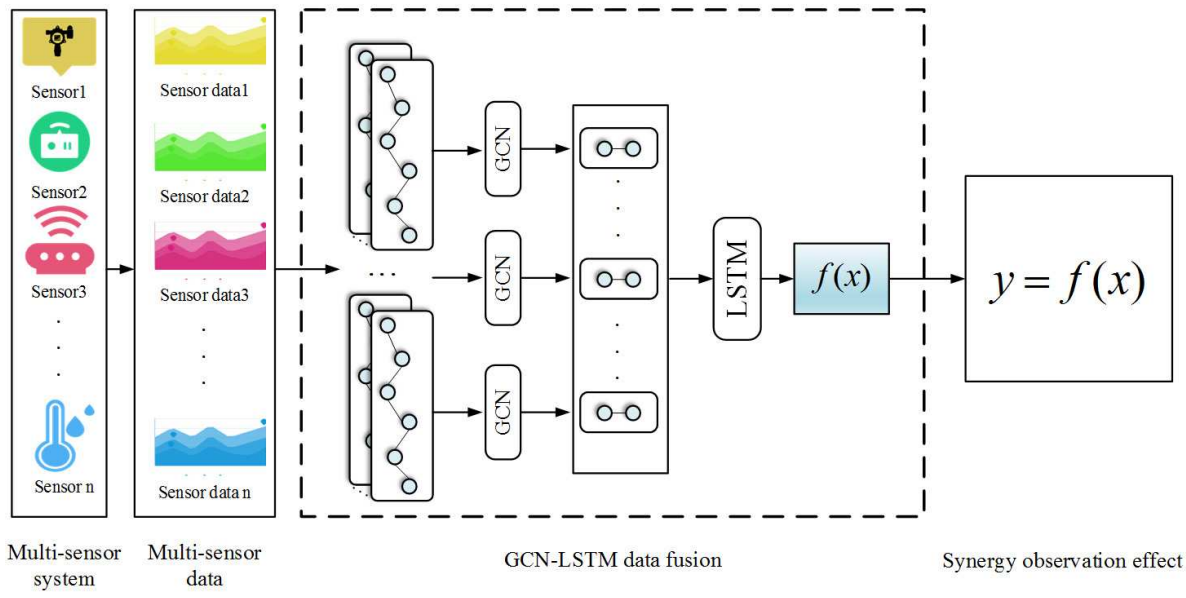


FIGURE 3. GCN-LSTM based multi-sensor system fusion framework

Based on the above analysis, we propose a model that fuses the spatially structured data and temporal structure of sensor data. The model is based on window moments and uses GCNs to extract spatial features between the input sensor data. Then, the GCN outputs of all window moments are combined into sequences, and the LSTM is used to evolve the GCN parameters to capture the dynamics of the sensor data sequences. Finally, the valid information in the input data is effectively extracted through a series of forgetting, remembering, updating, and direction propagation weight updating of the data sequences.

We extract and map the temporal and spatial features between sensor data by combining GCN and LSTM, and perform feature-level fusion to obtain consistent interpretation

and description of each sensor data to achieve multi-sensor data fusion, as shown in Figure 3.

3.2. **GCN-based spatial feature extraction and mapping.** In the deep learning model, CNN can extract Euclidean space features well and can be applied to the fusion of ordinary data. However, the composition of multi-sensors belongs to non-Euclidean space, and its data belongs to non-Euclidean data. Therefore, we use GCN neural network to extract spatial features between data from window moment sensor data and map to get the actual features, so as to get the fusion features of multi-sensors at window moment.

GCN contains multiple convolutional layers, each of which contains multiple convolutional kernels, each of which in turn contains multiple channels [22]. We assume that the convolutional layer of the lth layer is $G^{(l)}$, the convolutional kernel ith on that layer is $G_i^{(l)}$, and the channel jth on that kernel is $G_{i,j}^{(l)}$. $G^{(l)}$, $G_i^{(l)}$ and $G_{i,j}^{(l)}$ are shown in Equations (9)-(11).

$$G^{(l)} = \left[G_i^{(l)}\right]_{\alpha_l} \tag{9}$$

$$G_i^{(l)} = \left[G_{i,j}^{(l)}\right]_{\beta_l} \tag{10}$$

$$G_{i,j}^{(l)} = \sum_{\omega=0}^{\mu} \theta_{i,j,\omega}^{(l)} L^{\omega} \tag{11}$$

where μ is the number of polynomials; $\theta_{i,j,\omega}^{(l)}$ is the ωth term coefficients of the jth channel, which represents the weight influenced by the neighbor nodes of ω-order; α_l is the number of convolutional kernels in the lth convolutional layer; β_l is the number of convolutional kernel channels in the lth convolutional layer; L^{ω} is the Laplace matrix of order ω.

To quantify the cumulative effect of sensor data subject to different weight adjacency matrices $G_{i,j}^{(l)}$, weighted combinations $\theta_{i,j,z}^{(l)}$ and L^{ω} in GCN. We use the sensor data original features $h_c^{(0)}$ as input to the GCN, multiply the channel $G_{i,j}^{(l)}$ of each $G_i^{(l)}$ with the corresponding original feature $h_c^{(0)}$, sum the result of the multiplication and input the activation function, and finally obtain the higher-order spatial features $h_c^{(1)}$. After iterative convolution, more abstract higher-order spatial features are output, as shown in Equation (12).

$$h_c^{(l)} = \sigma\left(G^{(l)} \cdot h_c^{(l-1)}\right) \tag{12}$$

where $h_c^{(l)}$ is the spatial feature of layer l; $\sigma(\cdot)$ is the activation function.

In particular, to reduce the number of spatial feature mappings for sensor data and to improve the parameter training speed, we use a pooling layer to extract the convolved spatial features $h_c^{(l)}$, as shown in Equation (13).

$$h_c = \varphi\left(h_c^{(l)}\right) \tag{13}$$

where h_c is the pooled spatial feature; $\varphi(\cdot)$ is the pooling function.

Through the feedforward neural network, the GCN fully connected layer fully connects the pooled spatial features and maps them to obtain the mapping output $h_{fc}^{(l)}$, as shown in Equation (14).

$$h_{fc}^{(l)} = \sigma\left(W_{fc} h_{fc}^{(l-1)} + b_{fc}^{(l)}\right) \tag{14}$$

where $h_{fc}^{(l)}$ is the output of layer l fully connected layer, when $l = 0$, $h_{fc}^{(l)}$ is the input of layer l fully connected layer; $W_{fc}^{(l)}$ and $b_{fc}^{(l)}$ are the weights and bias terms of layer l fully connected layer, respectively.

After the l-layer fully connected layer, the mapping output A, i.e., the spatial fusion features of the sensor data at the window moment. In summary, the spatial feature extraction and mapping process based on GCN is shown in Figure 4.

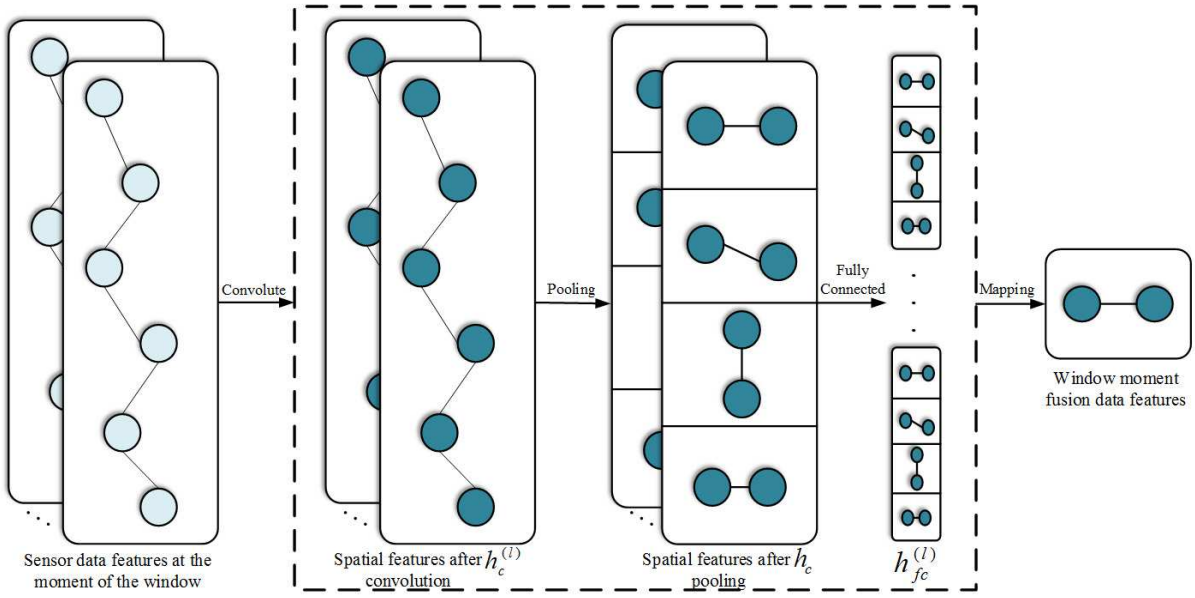


FIGURE 4. GCN-based spatial feature extraction and mapping

3.3. **LSTM-based temporal feature extraction and mapping.** In deep learning models, RNN is the classical method to extract time series. However, the usual RNN will make the gradient vanishing problem more and more serious, and the efficiency of temporal feature extraction will be greatly reduced. Therefore, we use LSTM to extract temporal features of sensor data, which can effectively solve the gradient vanishing problem and improve the efficiency of temporal feature extraction [23].

Based on Equations (3), (4), and (7), to distinguish from the convolution layer of GCN, we assume that the forgetting gate, input gate, and output gate of unit t are $\tilde{f}^{(t)}$, $\tilde{i}^{(t)}$ and $\tilde{o}^{(t)}$, respectively, the computational Equations (15)-(17).

$$\tilde{f}^{(t)} = \sigma\left(W_{\tilde{f}g}\left[h_{\tilde{f}}^{(t-1)}, h_n^{(t)}\right] + b_{\tilde{f}g}\right) \tag{15}$$

$$\tilde{i}^{(t)} = \sigma\left(W_{\tilde{i}g}\left[h_{\tilde{f}}^{(t-1)}, h_n^{(t)}\right] + b_{\tilde{i}g}\right) \tag{16}$$

$$\tilde{o}^{(t)} = \sigma\left(W_{\tilde{o}g}\left[h_{\tilde{f}}^{(t-1)}, h_n^{(t)}\right] + b_{\tilde{o}g}\right) \tag{17}$$

where $h_{\tilde{f}}^{(t-1)}$ is the significant temporal feature at time step $t-1$; $h_n^{(t)}$ is the original temporal feature at time step t; W and b are the weights and bias terms of the corresponding gates.

Specifically, we use the forgetting gate $\tilde{f}^{(t)}$ to combine the sensor temporal feature $h_{\tilde{f}}^{(t-1)}$ at the moment $t-1$ with the original sensor feature $h_n^{(t)}$ at the moment t. The cellular memory of the LSTM is controlled by the forgetting gate $\tilde{f}^{(t)}$ for temporal feature deletion, and the candidate temporal feature of the LSTM is controlled by the input gate $\tilde{i}^{(t)}$ for replenishment, so as to obtain the temporal feature at the moment t, which is calculated as shown in Equations (18) and (19).

$$T^{(t)} = \tilde{f}^{(t)} \cdot T^{(t-1)} + \tilde{i}^{(t)} \cdot \tilde{T}^{(t)} \tag{18}$$

$$\tilde{T}^{(t)} = \sigma\left(W_a\left[h_{\tilde{f}}^{(t-1)}, h_n^{(t)}\right] + b_a\right) \tag{19}$$

where $T^{(t)}$ is the temporal feature at time t; $\tilde{T}^{(t)}$ is the candidate temporal feature at time t; W_a and b_a are the weights and bias terms of the candidate temporal feature selection, respectively.

After the truncation and screening of the sensor temporal data features, the output of the important time features at the current moment is controlled by the output gate $\tilde{o}^{(t)}$. The time feature $T^{(t)}$ at the current t moment is input into the activation function and then outputted, and the calculation formula is shown in Equation (20).

$$h_{\tilde{o}}^{(t)} = \tilde{o} \cdot \sigma\left(T^{(t)}\right) \tag{20}$$

where $h_{\tilde{o}}^{(t)}$ is the sensor t temporal feature of the final output of output gate $\tilde{o}$.

The output important time features of the current time step will be input to the next unit to participate in the time feature extraction of the next time step, and after repeated operations until the last unit completes the time feature extraction [24].

The LSTM maps the significant features of the last time step $h_{\tilde{o}}^{(\delta)}$, and obtains the output of the mapping of the temporal features, i.e., the sensor data after feature fusion, as shown in Equation (21).

$$h = \sigma\left(W h_{\tilde{o}}^{(\delta)} + b\right) \tag{21}$$

where h is the output of the mapping; W and b are the weights and bias terms of the mapping, respectively.

In summary, the spatial feature extraction and mapping process based on LSTM is shown in Figure 5.

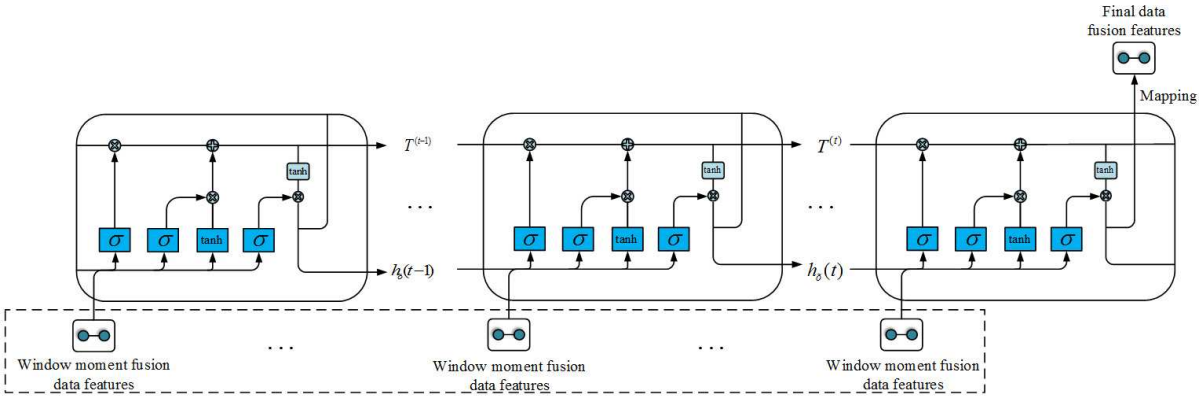


FIGURE 5. LSTM-based spatial feature extraction and mapping

3.4. **Model training.** The training process of the GCN-LSTM model for multi-sensor data fusion is shown in Figure 6. First, we divide the sensor data into a training set and a test set. Then, we input the training set into the GCN to extract the spatial features and mapping between sensor data, and extract the temporal features and mapping between sensor data in the LSTM to integrate the features of GCN-LSTM fusion to generate prediction data. Then, we iteratively train and judge whether the model training is completed, if not, we return to GCN and LSTM to continuously adjust the model structure, and if completed, we output the obtained model. Finally, we input the test set of sensors into the GCN-LSTM model to determine whether the accuracy meets the expectation, and if it is not completed, we still return to GCN and LSTM to keep adjusting the model structure, and if it is completed, we input the final model results.

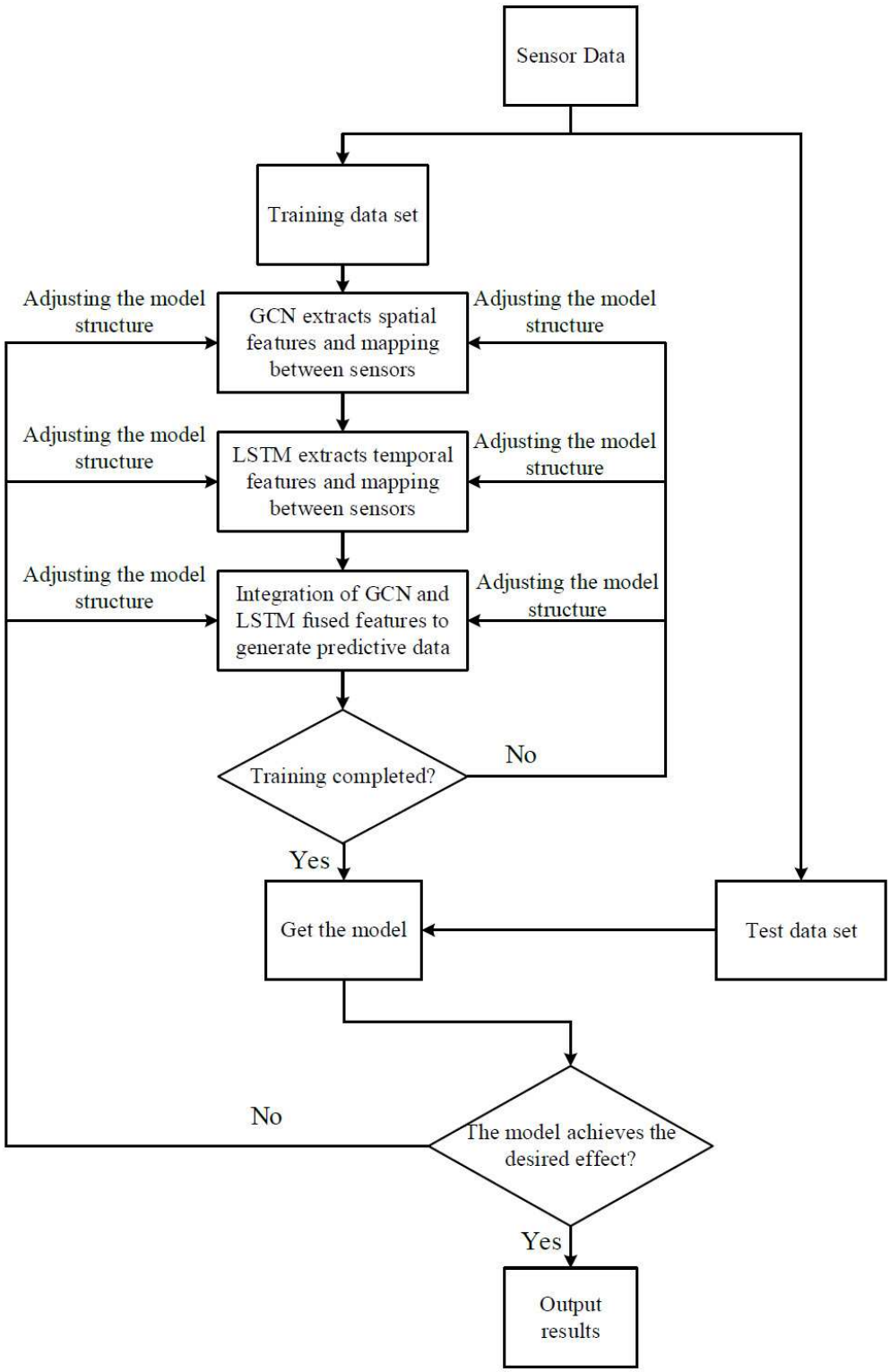


FIGURE 6. Model training process

4. **Simulation Experiments and Analysis.**

4.1. **Data set.** The data set for the simulation experiments is the real data collected from the Intel lab using 54 sensors during a month period, as shown in Figure 7. The sensed data in the lab included several features such as temperature, humidity, and light intensity [25]. The nodes of this sensor are based on a distributed architecture and have

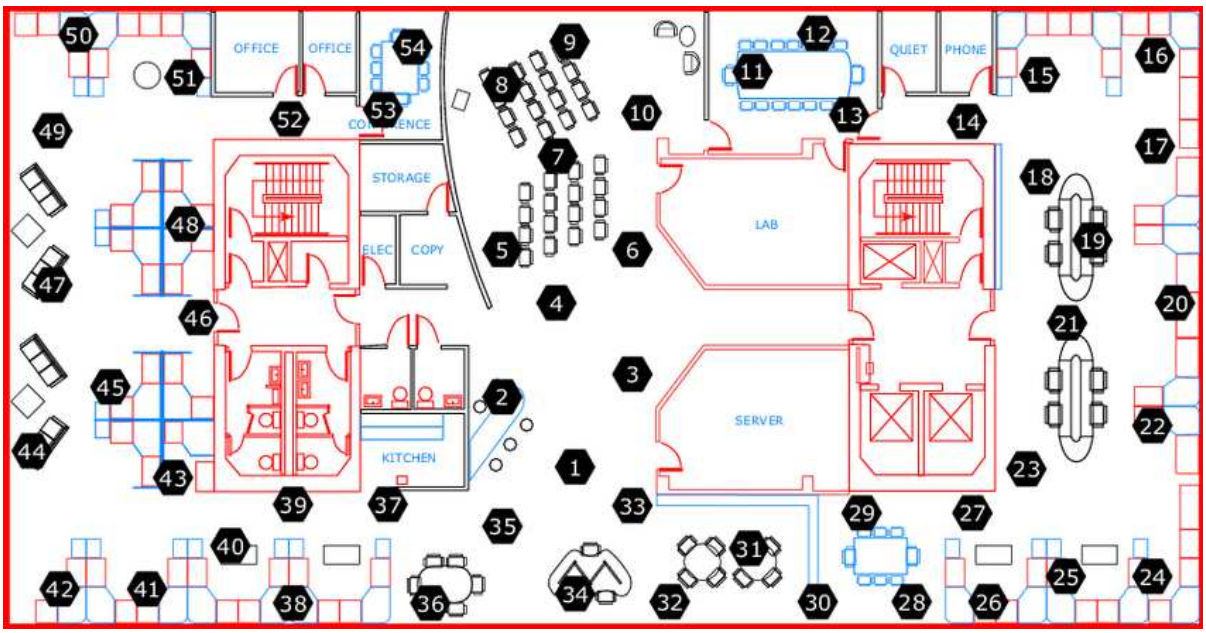


FIGURE 7. Intel labs sensor node distribution

a good basis for the clustering identification of IoT sensor devices [26]. Therefore, the collected multi-sensor data set is not only used for theoretical studies but also has strong practicality in industry and is used for more convincing experiments on sensor data fusion.

4.2. **Experimental setup.**

4.2.1. *Evaluation criteria.* In order to more intuitively express the deviation between the output results and the real results, we chose the mean absolute error (MAE), root mean square error (RMSE), and goodness-of-fit R^2 metrics to assess and evaluate the GCN-LSTM model [27]. MAE and RMSE are good metrics of model performance to minimize the loss function and to make the units of the results consistent with the data set and better described. R^2 is used to observe the degree of fit of the model predictions and has a good representation of the predicted data brought by the fused features. The equations for MAE, RMSE and R^2 are shown in Equations (22)-(24).

$$\mathrm{MAE} = \frac{1}{N}\sum_{n=1}^{N}|y_n - \hat{y}_n| \tag{22}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2} \tag{23}$$

$$\mathrm{R}^2 = 1 - \frac{\sum_{n=1}^{N}(y_n - \hat{y}_n)^2}{\sum_{n=1}^{N}(y_n - \bar{y})^2} \tag{24}$$

where N denotes the amount of data in the test set; y_n and $\hat{y}_n$ denote the true value of the nth instance and with the test, respectively; $\bar{y}$ denotes the average value of y. Both MAE and RMSE reflect the deviation of the predicted value from the true value, with smaller values indicating better model prediction. Larger R^2 means better model fit.

4.2.2. *Baselines.* To verify the effectiveness of GCN-LSTM for sensor data fusion, GCN-LSTM is compared with the following baselines. The baselines can be divided into three categories: general machine learning models (SVM), models based on temporal feature

extraction (GRU), and models based on temporal and non-Euclidean feature extraction (T-GCN, GCN-Seq2Seq).

SVM [28]: The existing data is analyzed to get the law that is not available in principle, and this law is used to make predictions for future data.

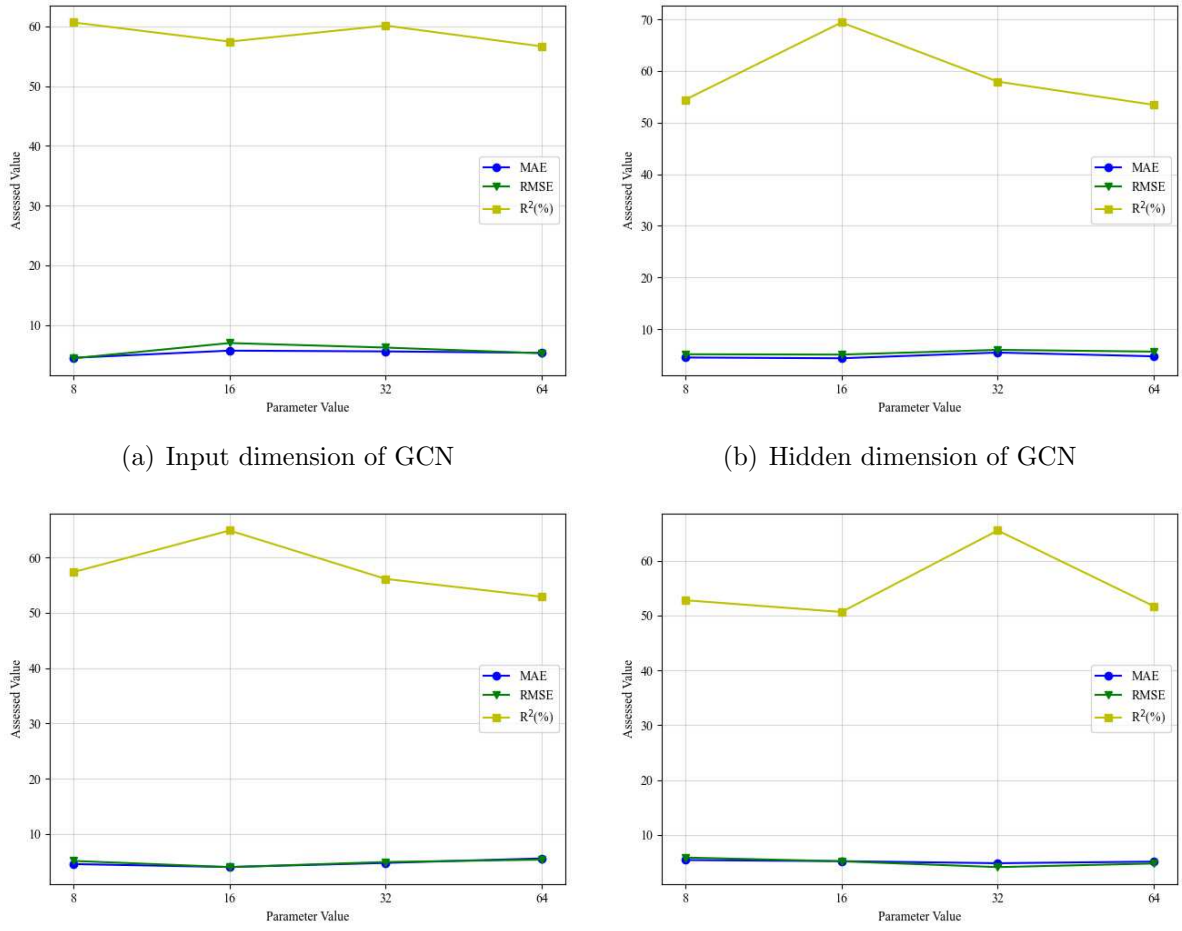
GRU [29]: A variation of LSTM, which synthesizes the forgetting gate and the input gate into a single update gate, and the rest is used to predict the future data through temporal features just like LSTM.

T-GCN [30]: Using GCN to obtain the spatial dependencies of the graph and GRU to learn the temporal dependencies of the nodes to predict future data by spatial and temporal features.

GCN-Seq2Seq [31]: Add two decoders to GCN and use the feature propagation function of the decoder to make predictions on future data.

4.2.3. *Parameter settings.* In terms of parameter settings, we pre-train the GCN-LSTM model to tune the parameters in order to better improve the accuracy of the model.

For the dimensional aspects of the model, the input dimensions of GCN are selected among $\{5, 10, 15, 20\}$, the hidden dimensions of GCN are selected among $\{8, 16, 32, 64\}$, the output dimensions of GCN are selected among $\{8, 16, 32, 64\}$, and the input dimensions of LSTM are selected among $\{8, 16, 32, 64\}$, and the results are shown in Figures 8(a)-8(d), respectively.



(a) Input dimension of GCN

(b) Hidden dimension of GCN

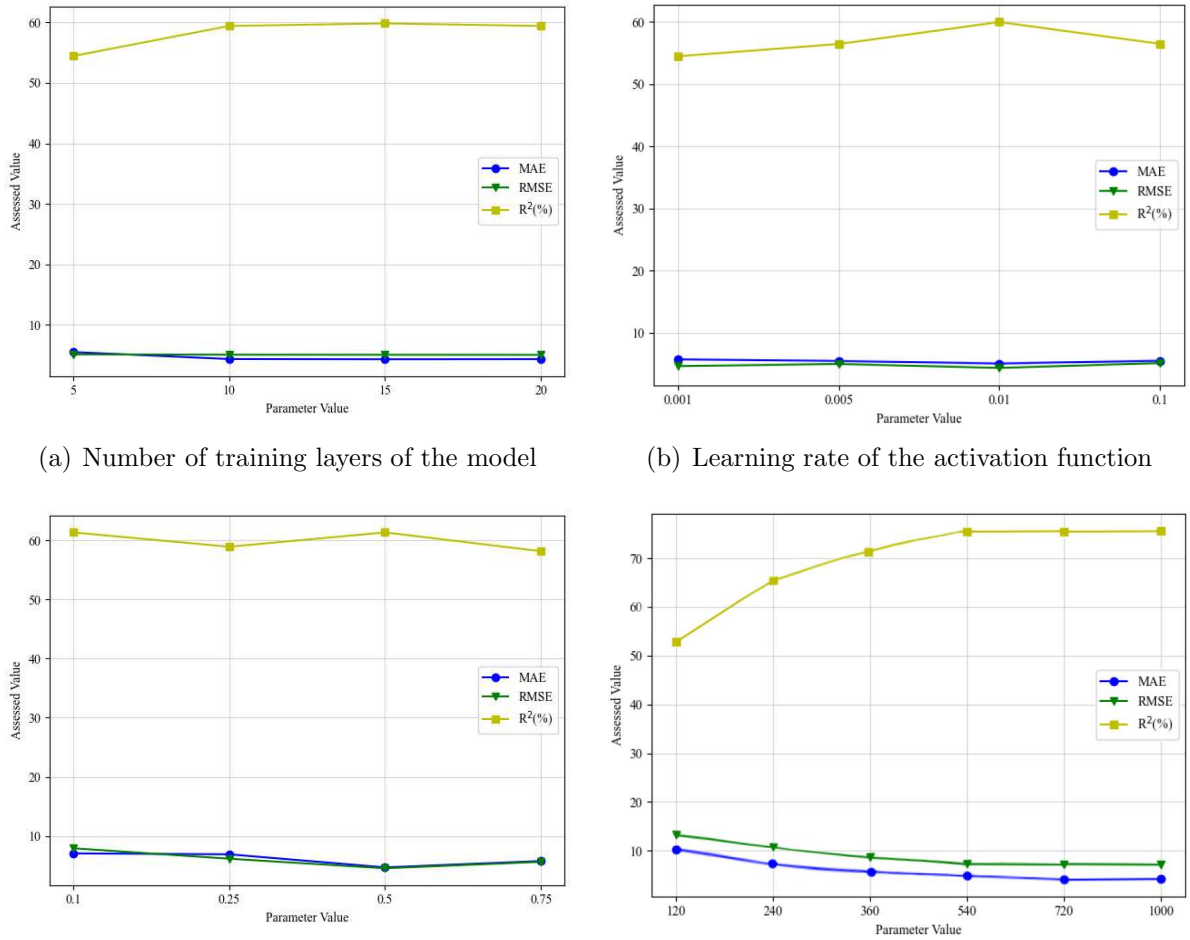
(c) Output dimension of GCN

(d) Input dimension of LSTM

FIGURE 8. Dimensional settings of the model

As can be seen from Figure 8, the MAE and RMSE are similar in terms of the choice of dimensionality, which may be due to the fact that dimensionality has little effect on the dispersion when the overall deviation is not large. The change in dimensionality has a greater effect on R^2, which may be due to the fact that the change in dimensionality leads to a larger change in the overall variance of the features.

For the other parameters of the model, the number of training layers of the model is chosen among $\{5, 10, 15, 20\}$, the learning factor of the activation function is chosen among $\{0.1, 0.01, 0.005, 0.001\}$, the ratio of retention in dropout is chosen among $\{0.1, 0.25, 0.5, 0.75\}$, and the number of epochs is initially set to 1000. The results are shown in Figures 9(a)-9(d), respectively.



(a) Number of training layers of the model

(b) Learning rate of the activation function

(c) Dropout retention rate

(d) Training epochs of the model

FIGURE 9. Other parameters of the model

From Figure 9, it can be seen that the variation of MAE and RMSE is relatively large in the selection of parameters for the internal structure of the model, which may be due to the fact that the variation of the internal structure directly affects the embedding of the features and thus the size of the metrics. The variation of the metrics of R^2 is relatively small, which may be due to the high overall fit of the model, thus ignoring the influence brought by the internal parameters of the model. In terms of overall training, each indicator leveled off after 540 epochs of the model.

According to Figure 8 and Figure 9, the parameters are chosen when MAE and RMSE are small, and R^2 is large. Also, the number of training epochs is chosen to be 540 to reduce the computational cost. The configuration of each parameter is shown in Table 1.

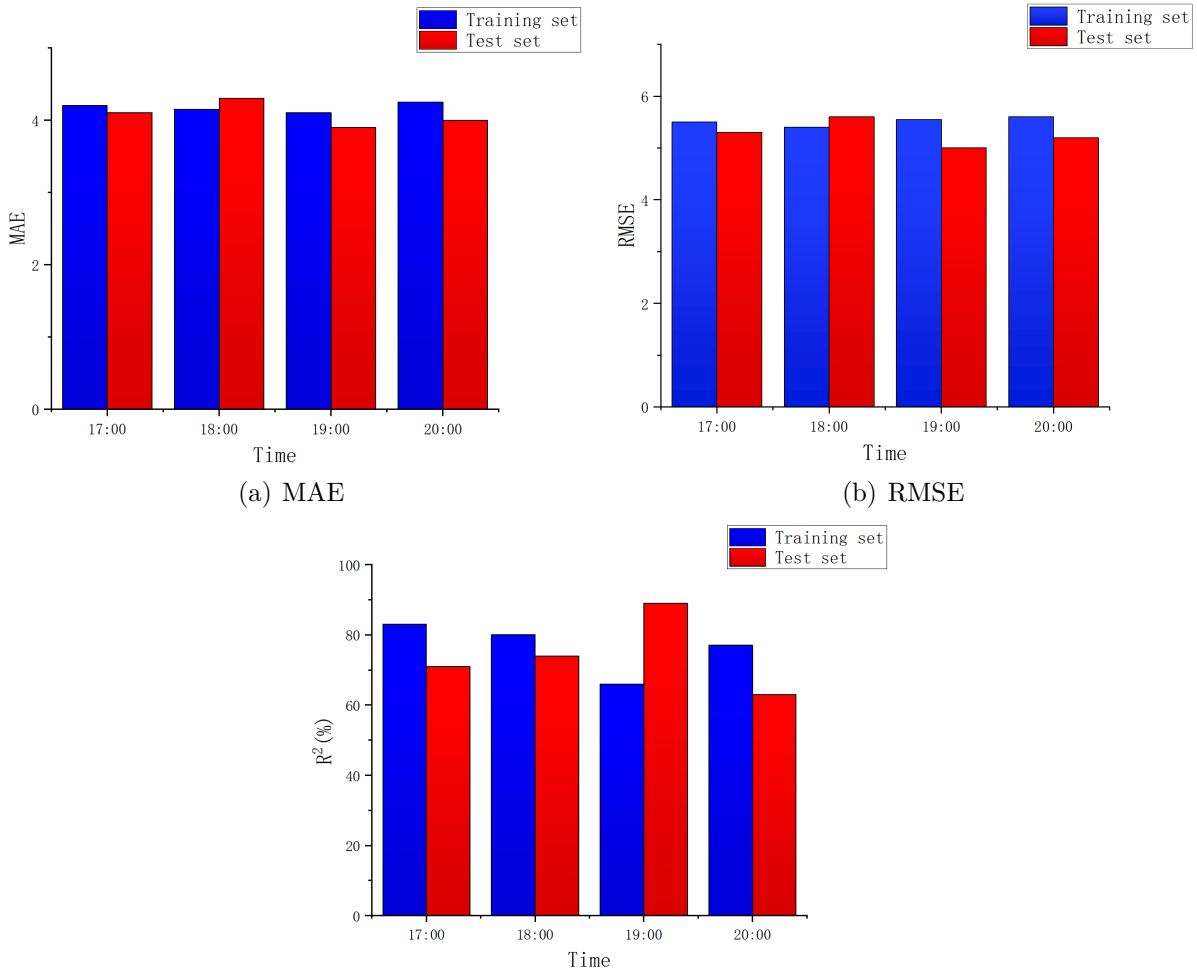
Table 1. Other parameters of the model

Parameter name	Parameter meaning	Parameter value
gcn_in	Input dimension of GCN	5
gcn_hid	Hidden dimension of GCN	16
gcn_out	Output dimension of GCN	16
lstm_in	Input dimension of LSTM	32
l	Number of training layers	10
α	Learning rate within the activation function	0.01
β	Dropout retention rate	0.5
n	Training epochs	540

4.3. **Experimental results and comparison.**

4.3.1. *Overall performance.* The overall simulation experiments are performed on 54 sensor node data from Intel labs, and the simulation data is extracted from each sensor node, which contains 3000 consecutive data sequences. To ensure the feasibility and fairness of the experimental comparison, the data sequences are divided into training and test sets in a ratio of 3 : 1. We use the past 250 historical adoption data as the input to the model and output 1 prediction per target. Since the first 250 sampled data from each node are used to train the GCN-LSTM model, we only perform statistical analysis of the data after 250 sampled points. We fused the sampled data through the GCN-LSTM model used for sensor data fusion and performed sensor node data prediction to determine the overall performance capability of the model. Specifically, we randomly selected four times with the same interval to predict all the sampled sensor data by the GCN-LSTM model, calculated the error between the real data collected by the nodes and evaluated the overall performance of the GCN-LSTM model using MAE, RMSE and R^2 metrics, and the results are shown in Figure 10. From the results of the overall test, it can be obtained that the MAE of GCN-LSTM is lower than 5, RMSE is lower than 6, and R^2 is higher than 60% on both the training and test sets, which indicates that the model is effective and performs well for data fusion on the whole sensor system.

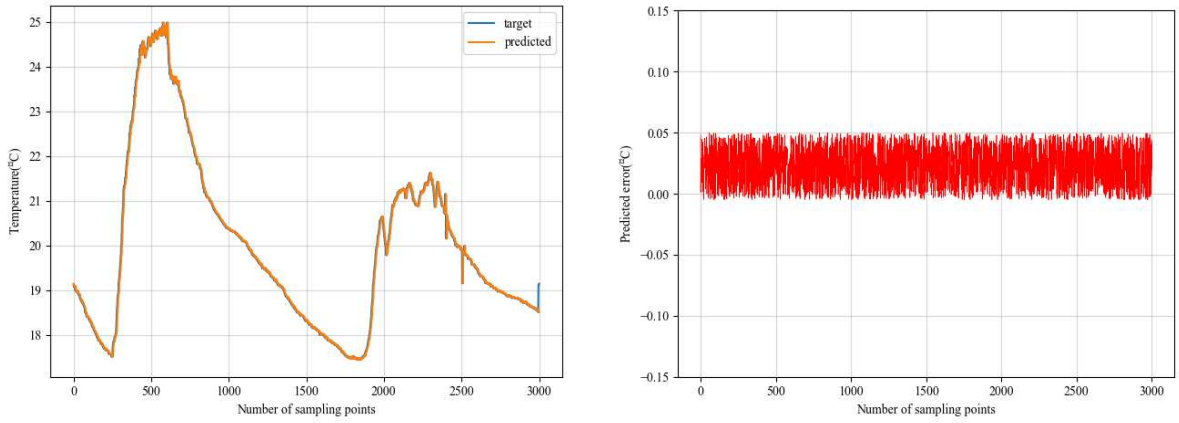
4.3.2. *Individual performance.* The overall performance of the GCN-LSTM model represents the average of the errors of all nodes within the predicted sensor system, but there are differences in the individual performance of each sensor node, i.e., the errors between the real data collected by each node and the fused predicted data differ, and the individual performance of the GCN-LSTM needs to be analyzed by the errors of each node. Therefore, we collected node 9 temperature sensor data and node 10 temperature sensor data in Intel labs, fused the data features and continuously predicted the future data by the GCN-LSTM model, and also calculated the error between the predicted data and the real data of the sensors, and the results are shown in Figure 11 and Figure 12. From the prediction results, it can be seen that the predicted data curve of GCN-LSTM basically fits with the actual data curve. The time series itself has strong time correlation, and the prediction mechanism of this model has been able to achieve a high accuracy rate, which is very suitable for data fusion of multi-sensor systems.



(a) MAE (b) RMSE

(c) R^2

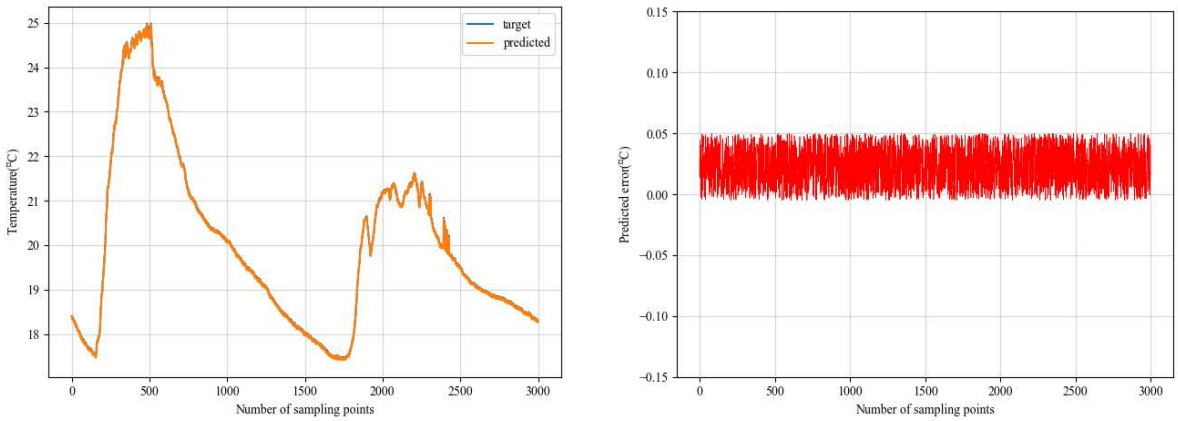
FIGURE 10. Overall performance



(a) Predicted results (b) Error situation

FIGURE 11. Node 9 temperature prediction results and error situation

4.3.3. *Model comparison.* To compare the GCN-LSTM model with other models, we constructed a collection of comparison models using the baseline shown in 4.2.2, which includes four models, SVM, GRU, T-GCN, and GCN-Seq2Seq. The overall performance of



(a) Predicted results (b) Error situation

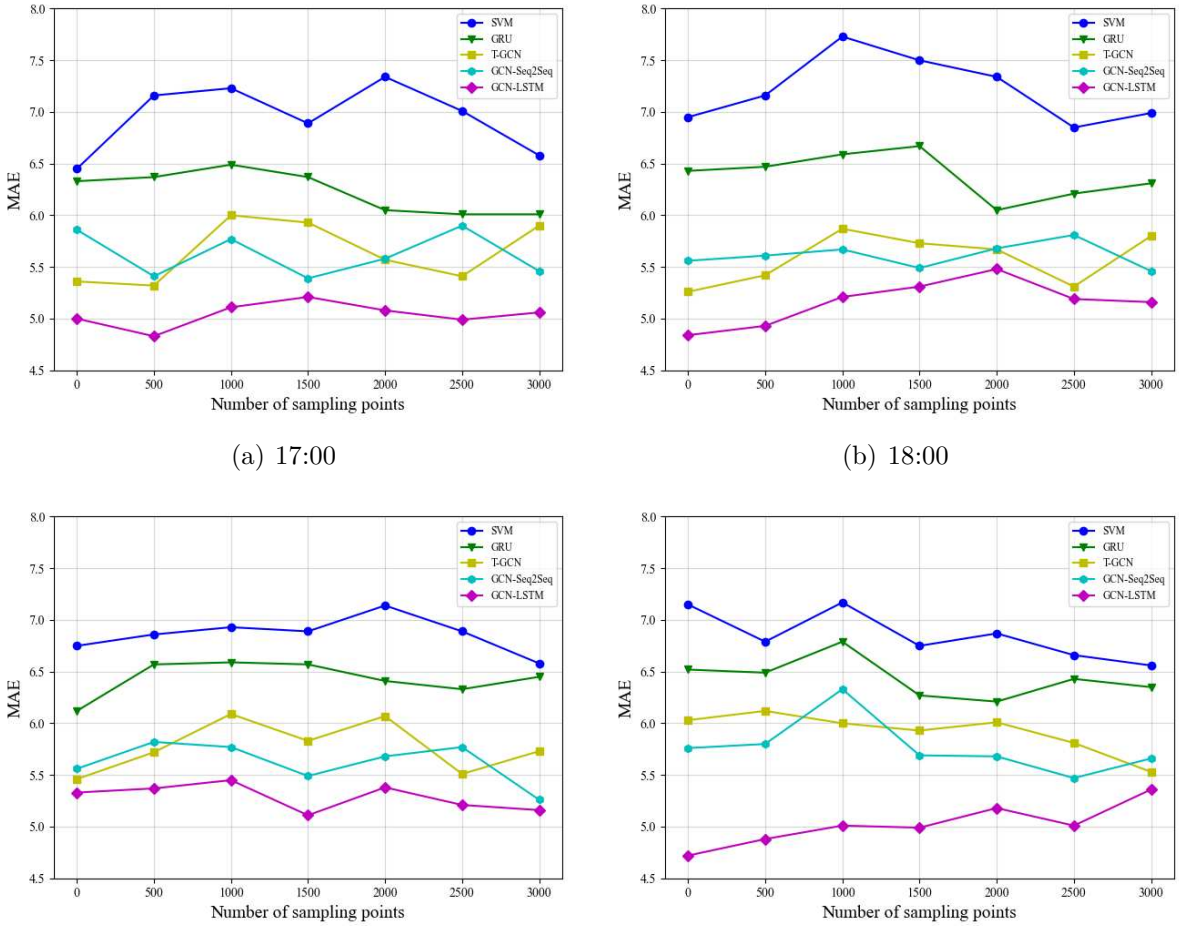
FIGURE 12. Node 10 temperature prediction results and error situation

TABLE 2. Comparison of the overall performance of different models

Model name	MAE				RMSE				R^2/%			
	17:00	18:00	19:00	20:00	17:00	18:00	19:00	20:00	17:00	18:00	19:00	20:00
SVM	6.73	6.20	5.37	4.37	8.09	7.98	6.44	7.02	33.98	38.30	30.97	33.82
GRU	6.41	5.89	4.61	4.73	6.84	6.58	6.74	7.52	54.19	60.22	66.74	60.72
T-GCN	5.12	5.37	4.57	4.32	6.78	6.36	6.66	7.44	55.95	64.71	67.97	59.34
GCN-Seq2Seq	5.71	5.34	5.32	4.88	6.94	6.00	5.57	6.23	56.43	63.81	68.16	60.65
GCN-LSTM	**4.17**	**4.44**	**3.98**	**4.25**	**5.33**	**5.61**	**5.27**	**5.13**	**71.88**	**74.24**	**89.43**	**63.58**

each model on the training set was evaluated using three evaluation metrics, MAE, RMSE, and R^2, each based on the mean value of the error between the node value predicted by model fusion and the actual node value of the sensor. The specific results are shown in Table 2, where the bolded results are the optimal solutions and the underlined results are the suboptimal solutions. From Table 2, it can be obtained that the SVM model has the worst three metrics, which is due to the fact that there is no general machine learning model that does not predict for the features of the sensor data, resulting in poor results; the GRU model, although the effect is better than the general machine learning model, does not achieve particularly good results, which is due to the fact that such models only consider the temporal features and ignore the spatial features among the sensor data; the T-GCN model and GCN-Seq2Seq model have been significantly better than the first two, which is due to their ability to extract temporal and spatial features and fit the features of the sensor data well; the GCN-LSTM model we used achieves the optimal solution in all three metrics, which is due to the model's ability to target temporal and spatial feature extraction, effectively reducing the prediction error and obtaining better sequence results.

To investigate the individual performance of each model on the test set, we still collected temperature data from sensor node 9 and sensor node 10, and set the same time interval as the overall performance of each model to evaluate the individual performance of each model on the test set using the MAE metric. Specifically, we used the same 3000 consecutive sequences on both sensor nodes, and the value of MAE was recorded every



(a) 17:00 (b) 18:00

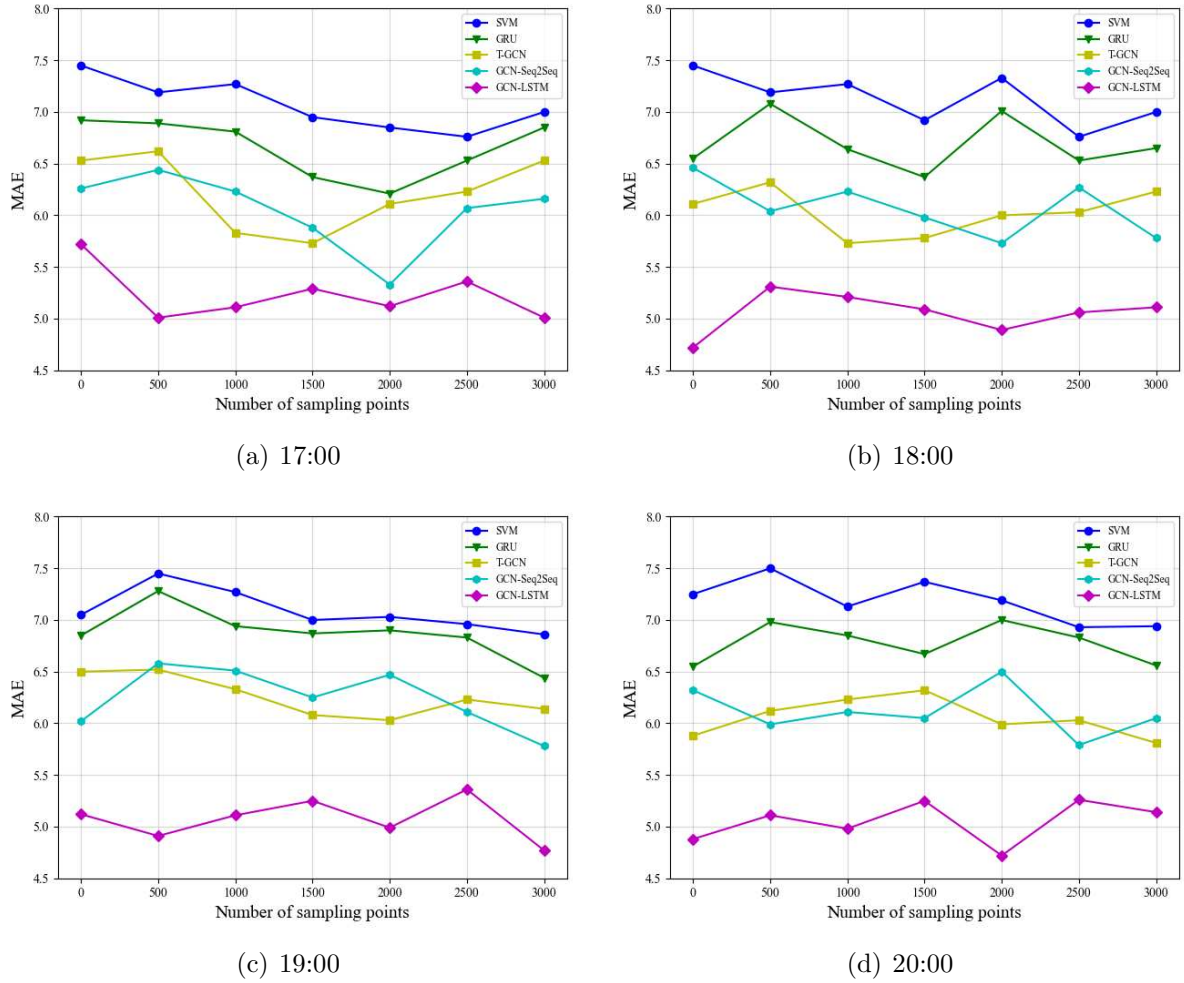
(c) 19:00 (d) 20:00

FIGURE 13. Performance of each model at node 9

500 sequence points to compare the performance capability of each model, as shown in Figure 13 and Figure 14.

From Figure 13 and Figure 14, it can be seen that in terms of individual performance, the MAE value of the SVM model is the highest and the MAE value of the GRU model is the second highest, which is because both of them do not extract features well for sensor data; the MAE values of the T-GCN model and the GCN-Seq2Seq model are high and low, which may be due to the fact that both of them have different temporal extraction ability and spatial extraction ability for sensor data; the MAE value of the GCN-LSTM is the lowest, which proves that the individual performance ability is also excellent, which is attributed to the temporal and spatial extraction ability for sensor data.

5. **Conclusions.** In this study, a hybrid algorithm based on GCN and LSTM is proposed to establish a model for multi-sensor data fusion. Among them, the graph neural network is applied to extracting non-Euclidean spatial features, which solves the problem of difficult fusion of heterogeneous data caused by the difference of data types; the feature extraction of time series by LSTM solves the problem of gradient disappearance. The experimental results show that the proposed GCN-LSTM fusion algorithm has better accuracy and performance than other algorithms in predicting the future data direction of nodes in the environment of multi-sensor nodes.



(a) 17:00 (b) 18:00

(c) 19:00 (d) 20:00

FIGURE 14. Performance of each model at node 10

Acknowledgment. This work is partially supported by "National Natural Science Foundation of China, No. 61762031", "Science and Technology Major Project of Guangxi Province, No. AA19046004", "Natural Science Foundation of Guangxi, No. 2021JJA170-130" and "Innovation Project of Guangxi Graduate Education, No. YCSW2022326". The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] Z. Liu, G. Xiao, H. Liu et al., Multi-sensor measurement and data fusion, *IEEE Instrumentation & Measurement Magazine*, vol.25, no.1, pp.28-36, 2022.

[2] Anderies, B. A. Jabar, R. Yunanda and A. A. S. Gunawan, The development of a smart door decision system, based on PIR sensor, embedded face recognition and server request using TTGO ESP 32, *ICIC Express Letters, Part B: Applications*, vol.12, no.10, pp.965-970, 2021.

[3] S. A. Kashinath, S. A. Mostafa, A. Mustapha et al., Review of data fusion methods for real-time and multi-sensor traffic flow analysis, *IEEE Access*, vol.9, pp.51258-51276, 2021.

[4] S. Smaiah, R. Sadoun, A. Elouardi et al., A practical approach for high precision reconstruction of a motorcycle trajectory using a low-cost multi-sensor system, *Sensors*, vol.18, no.7, pp.2282-2299, 2018.

[5] R. Gutiérrez, V. Ramperez, H. Paggi et al., On the use of information fusion techniques to improve information quality: Taxonomy, opportunities and challenges, *Information Fusion*, vol.78, pp.102-137, 2022.

[6] S. Gao, B. Zhang, J. Sun et al., Research and application of a multi-sensor data fusion algorithm based on an improved back propagation neural network by particle swarm optimization, *Journal of Nonlinear and Convex Analysis*, vol.21, no.7, pp.1497-1510, 2020.
[7] Y. Jiao, T. Lou, X. Wang et al., The KF-SVM-based fusion method for multi sensor uncertain system with correlated noise, *Journal of Intelligent & Fuzzy Systems*, pp.1-11, 2021.
[8] W. Qi, S. E. Ovur, Z. Li et al., Multi-sensor guided hand gesture recognition for a teleoperated robot using a recurrent neural network, *IEEE Robotics and Automation Letters*, vol.6, no.3, pp.6039-6045, 2021.
[9] J. J. A. Lima, L. F. Maldaner and J. P. Molin, Sensor fusion with NARX neural network to predict the mass flow in a sugarcane harvester, *Sensors*, vol.21, no.13, pp.4530-4543, 2021.
[10] N. A. Asif, Y. Sarker, R. K. Chakrabortty et al., Graph neural network: A comprehensive review on non-Euclidean space, *IEEE Access*, vol.9, pp.60588-60606, 2021.
[11] J. Bruna, W. Zaremba, A. Szlam et al., Spectral networks and locally connected networks on graphs, *arXiv Preprint*, arXiv: 1312.6203, 2013.
[12] Y. Yu, X. Si, C. Hu et al., A review of recurrent neural networks: LSTM cells and network architectures, *Neural Computation*, vol.31, no.7, pp.1235-1270, 2019.
[13] A. A. Aguileta, R. F. Brena, O. Mayora et al., Multi-sensor fusion for activity recognition – A survey, *Sensors*, vol.19, no.17, pp.3808-3848, 2019.
[14] X. Bi, Multimodal sensor collaborative information sensing technology, in *Environmental Perception Technology for Unmanned Systems*, Singapore, Springer, 2021.
[15] X. Ji and X. Cheng, An adaptive multisensor image fusion method based on monogenic features, *IEEE Sensors Journal*, vol.21, no.14, pp.15598-15606, 2020.
[16] Y. Wu and J. Feng, Development and application of artificial neural network, *Wireless Personal Communications*, vol.102, no.2, pp.1645-1656, 2018.
[17] H. Zhang, G. Lu, M. Zhan et al., Semi-supervised classification of graph convolutional networks with Laplacian rank constraints, *Neural Processing Letters*, pp.1-12, 2021.
[18] J. He and M. Cheng, Graph convolutional neural networks for power line outage identification, *2020 25th International Conference on Pattern Recognition (ICPR)*, pp.4198-4205, 2021.
[19] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Computation*, vol.9, no.8, pp.1735-1780, 1997.
[20] A. Fathalla, K. Li, A. Salah et al., An LSTM-based distributed scheme for data transmission reduction of IoT systems, *Neurocomputing*, vol.485, pp.166-180, 2022.
[21] W. Liao, B. Zeng, J. Liu et al., Multi-level graph neural network for text sentiment analysis, *Computers & Electrical Engineering*, vol.92, DOI: 10.1016/j.compeleceng.2021.107096, 2021.
[22] X. Wang, M. Zhu, D. Bo et al., AM-GCN: Adaptive multi-channel graph convolutional networks, *Proc. of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp.1243-1253, 2020.
[23] C. Qin, G. Shi, J. Tao et al., Precise cutterhead torque prediction for shield tunneling machines using a novel hybrid deep neural network, *Mechanical Systems and Signal Processing*, vol.151, DOI: 10.1016/j.ymssp.2020.107386, 2021.
[24] F. J. Ordóñez and D. Roggen, Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition, *Sensors*, vol.16, no.1, pp.115-139, 2016.
[25] *Intel Berkeley Research Lab*, http://db.csail.mit.edu/labdata/labdata.html, 2004.
[26] J. Liu, Y. Sun, F. Xu et al., IIS: Intelligent identification scheme of massive IoT devices, *2021 IEEE the 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp.1623-1626, 2021.
[27] R. A. Collenteur, How good is your model fit? Weighted goodness-of-fit metrics for irregular time series, *Ground Water*, vol.59, no.4, pp.474-478, 2021.
[28] T. Meng, X. Jing, Z. Yan et al., A survey on machine learning for data fusion, *Information Fusion*, vol.57, pp.115-129, 2020.
[29] J. Liu, T. Li, P. Xie et al., Urban big data fusion based on deep learning: An overview, *Information Fusion*, vol.53, pp.123-133, 2020.
[30] G. Jin, Y. Cui, L. Zeng et al., Urban ride-hailing demand prediction with multiple spatio-temporal information fusion network, *Transportation Research Part C: Emerging Technologies*, vol.117, DOI: 10.1016/j.trc.2020.102665, 2020.
[31] Y. Chen, P. Yang, C. Ye et al., GCN-Seq2Seq: A spatio-temporal feature-fused model for surface water quality prediction, *2021 5th International Conference on Computer Science and Artificial Intelligence*, pp.317-322, 2021.

Author Biography

Bohuai Xiao is a postgraduate student in the Department of Information Science and Engineering of Guilin University of Technology. His main research interests include deep learning, graph neural network and recommendation system.

Xiaolan Xie obtained her Ph.D. degree in mechanical manufacturing and its automation from Xidian University, China in 2009.

Prof. Xie is currently the dean of the Department of Information Science and Engineering at Guilin University of Technology. Her main research interests include cloud computing, big data and manufacturing information. She has published more than 100 scientific papers, including more than 50 SCI/EI indexed papers. She is hosting some research projects funded from the National Natural Science Foundation of China.

Chengyong Yang obtained his M.S. degree in software engineering from University of Electronic Science and Technology.

Mr. Yang is currently the deputy director of the Network and Information Center at Guilin University of Technology. He is mainly engaged in cloud computing, Internet of Things, big data processing and recommendation algorithm research.