Mechanical Systems and Signal Processing 156 (2021) 107650

Contents lists available at ScienceDirect

Mechanical Systems and Signal Processing

journal homepage: www.elsevier.com/locate/ymssp

Collaborative deep learning framework for fault diagnosis in distributed complex systems

Haoxiang Wang [a], Chao Liu [a,b,*], Dongxiang Jiang [a,c], Zhanhong Jiang [d]

[a] Department of Energy and Power Engineering, Tsinghua University, Beijing 100084, China
[b] Key laboratory for Thermal Science and Power Engineering of Ministry of Education, Tsinghua University, Beijing 100084, China
[c] State Key Laboratory of Control and Simulation of Power System and Generation Equipment, Tsinghua University, Beijing 100084, China
[d] Johnson Controls, 507 East Michigan St, Milwaukee, WI 53202, USA

ARTICLE INFO

Article history:
Received 4 August 2020
Received in revised form 11 December 2020
Accepted 11 January 2021
Available online 18 February 2021

Keywords:
Fault diagnosis
Distributed complex systems
Collaborative deep learning
Privacy preserving

ABSTRACT

In distributed complex systems, condition monitoring and fault diagnosis have received considerable attention, especially for recent developments of data-driven methods with deep learning structures, which greatly enhance the performance as of superior representation capacity over big data. To apply these methods, massive data needs to be collected from distributed systems, requiring high costs for data transmission and causing more and more concerns on privacy issues. For the naturally distributed data in such scenario, this work presents a novel collaborative deep learning framework with the idea that the features, as representations of data, can be transmitted through latent parameters of deep learning structure while the raw data won't be shared in the distributed network. Based on the collaborative learning setup, the proposed framework adopts a secure communicating strategy with no need of transmitting raw data, and obtains a consensus for distributed deep learning models that can be geographically located. To validate the proposed scheme, four case studies are carried out and the results show that it is able to improve the diagnosis accuracy compared with local learning models. Also, it is robust and adaptive for diagnosis problems with data that is imbalanced or from different distributions.

1. Introduction

In distributed complex systems, such as mechanical systems (bearing, gearbox, rotor system, which are widely adopted in diverse applications), energy systems (wind, solar, thermal power plants, extensively installed depending on the circulation of energy resources), condition monitoring and fault diagnosis have received considerable attention. The data collected from the aforementioned complex systems is with a common characteristic that the data is spatially distributed, and it is usually stored in local data centers (e.g., a wind farm manages the data from tens or hundreds of wind turbines), which probably belongs to different owners. Traditionally, data is transmitted to the central data management center on the purpose of storage and data mining within the same owner. However, data sharing with different owners probably causes concerns, such as security, privacy, etc [1]. Also, managing a single data center for the whole system may not be efficient in terms of the cost, security and availability [2]. Taking wind farms in China as an example [3], data collection and condition monitoring centers are set up in various wind power enrichment locations for daily operation and maintenance. Within the local data centers

* Corresponding author at: Department of Energy and Power Engineering, Tsinghua University, Beijing 100084, China.
E-mail address: cliu5@tsinghua.edu.cn (C. Liu).

https://doi.org/10.1016/j.ymssp.2021.107650
0888-3270/

belonging to the same owner, different wind turbines have the same type of data [4]. As the data centers are affiliated with different companies, data sharing is not allowed due to the privacy and the security concerns. However, data sharing is essential for big data analytics, especially for distributed systems. The reason is that condition monitoring data probably has limited variety of faults and diversity of operating conditions for a local data center. Therefore, forming a diagnosis model needs more data from other data centers.

In the recent year, deep learning framework in fault diagnosis has attracted much attention. These kinds of models distinguish themselves with high accuracy [5–7]. But the requirement of the variety and volume of data for a diagnosis framework is more significant for deep learning structures.

Intensive researches have been conducted in the community with the characteristics of deep learning on highly nonlinear representing power between the inputs and outputs, automatic feature extraction in an end-to-end manner, and improved performance, robustness and versatility over shallow methods[8–11].

Enjoying nonlinear modeling capacity between the inputs and outputs, those overparameterized deep learning approaches highly rely on the amount of data, to solve optimization problems and avoid overfitting [12]. For example, the diagnosis accuracy decreases with the reducing size of available data in deep long-short term memory (LSTM) networks and convolutional neural networks (CNNs) [13].

Therefore, the gap between the data requirement for deep learning methods and the data distributivity for complex systems is one of the major challenges.

To collect enough data for deep learning structures while maintaining privacy, training the models in distributed way has been attracting more and more attention, where Federated Averaging and its variants are considered to be a great method [14]. The intention of the federated learning is to deploy the algorithms among the edge computing resources such as smart phones and idle computers, where the data is available from individual agents while model parameters of deep learning structures are managed in a central server (master-slave architecture). Such a method is widely used in different fields like medical and IoT [15,16]. However, the distributed learning schemes need more discussion in the task of fault diagnosis, where both source and demand of data have wide distribution.

In this context, a distributed learning scheme, noted as collaborative deep learning, is presented in this work to address: (i) the scenario of deep learning in distributed complex systems, where the data and computing power are both geographically distributed, for the purpose of bridging the gap between the data dispersity for complex mechanical systems and the data requirement for deep learning models, (ii) data sharing with privacy preserving, where the raw data can not be transmitted, and the information only for fault diagnosis represented by the model parameters will be transmitted, (iii) utilization of all the computing resources that are geographically dispersed, where the connecting and communicating strategy is required to determine what and when to communicate in the distributed learning framework.

As discussed above, data is massively collected in a distributed manner and a suitable distributed learning scheme design is important for improving accuracy of data-driven fault diagnosis. Various distributed learning schemes were proposed recently, including Federated Averaging we mentioned above and its several variants [17–20]. However, actually most of them focus on high Performance Computing (HPC) clusters or Edge Computing units to address the limit of computing resources for each unit, which requires a central server. But in our scenario, a central server means a possible loophole due to the manipulation of centre's privilege. Therefore, in our applications of distributed industrial systems, a peer-to-peer architecture is more appropriate in terms of the decentralized model and its parameters. There are still some decentralized methods like gossip algorithm [21]. But this method utilizes the probability communication, which is not stable compared with the proposed collaborative deep learning framework for fault diagnosis in our paper.

The specific contributions of this work are as follows.

- The collaborative deep learning framework is presented for fault diagnosis in distributed complex (mechanical, energy, etc.) systems, which is privacy preserving with no need of raw data sharing. Towards parameter communication of deep model architectures among agents in the distributed network, the proposed framework is shown to be robust and adaptive for diagnosis problems with data that is even imbalanced or from different distributions (collected in diverse components, e.g., different bearings or devices).
- Based on synchronous consensus-based distributed stochastic gradient descent (CDSGD), the consensus-based distributed Adam (CDAdam) algorithm is proposed for the collaborative learning framework in practical scenarios. By adding a communication layer, the presented framework enjoys the privileges of parallel computing and easy implementation. Additionally, the proposed CDAdam delivers us one quite useful insight in the consensus-based distributed adaptive gradient descent algorithms, which can embrace other popular optimizers, such as AdaGrad, RMSProp, etc., for other applications.
- The proposed framework is validated with four case studies, and the presented framework is shown to be able to improve the diagnosis accuracy without transmitting raw data between distributed complex systems. Especially for the imbalanced data among the agents of the distributed network, the presented framework boosts the classification accuracy among the agents (close to the central setting, but without raw data transmission.)

The significance of our proposed method mainly lies in three folds. First, in the era of big data, deep learning models have become the state-of-the-art that typically requires a large amount of data in engineering problems such as fault diagnosis. The rapid development of computing devices and communication technologies also enables the distributed computing,

which correspondingly requires effective distributed learning algorithms. Second, compared to the traditional methods, data privacy and distributions are two key issues that now have received considerable attention from the community. Generically, they are existing in almost all engineering systems and so our collaborative learning framework provides the first attempt to solve the problems and maintains the favorably comparable performance as in the centralized systems for the fault diagnosis. Third, the development of the distributed algorithm can be extensively adapted in different applications with other core methods. Our communication layer design and field implementation workflow are able to be adopted in other time series problems, e.g., forecasting and prediction.

The rest of the paper is outlined as follows. In Section 2, we provide preliminaries. In Section 3 we present our collaborative learning algorithm for distributed renewable energy systems. Experimental results are shown in Section 4 and concluding remarks are given in Section 5.

2. Preliminaries and background

2.1. Deep learning for fault diagnosis

This work is focused on deep learning algorithms with the massive data collected from distributed mechanical and energy systems, for fault diagnosis, prognosis and health management. In the recent year, deep learning framework in fault diagnosis has attracted much attention. For an instance, Wang et al. [22] and Chen et al. [23] adopted the deep confidence network to achieve fault diagnosis of rolling bearings and gearboxes, and the robustness and accuracy are comparable to the state-of-the-art method. Janssen et al. [24] applied convolutional neural network (CNN) to solve two traditional problems including outer ring raceway failure and lubrication performance degradation in rotating machinery. Nasser et al. [25] constructed a recurrent neural network model to analyze the operation behavior of the wind power system, which can finally perform fault identification in a short time, with a very low false alarm rate. Besides the literature above, more convolutional neural networks (CNN) based diagnosis models [26], recurrent neural networks (RNN) based diagnosis models [27] and long-short term (LSTM) based models [28]. In this work, we mainly focus on the long-short term (LSTM) based models. These model could extract the long-term relationship in the time series fault data compared with other models and it could derive a more accurate result in the fault diagnosis task.

However, these deep learning methods need a large amount of data to achieve a good performance. Considering the data is usually scattered in distributed industrial systems, this work intends to develop a method to provide the deep learning networks with more data in a distributed learning manner. With more available data, the deep learning networks could enjoy the privileges of massive data for better solving the optimization problem and improving the diagnosis accuracy.

2.2. Long short-term memory (LSTM) networks for fault diagnosis

The deep learning model this work applied is the Long Short-Term Memory (LSTM) model [29], which is mainly applied to the analysis of time series data, including Natural Language Processing. The main idea of LSTM is to update the hidden state h_t by the continuous input of time series to reflect the features embedded.

However, due to a large number of back propagation iterations, only a single feature extraction design will encounter the problem of vanishing gradient in practical applications. Therefore, in order to ensure a better gradient flow, LSTM model has been applied, which introduces two mechanisms of memory and forgetting, providing a highway for gradient back propagation, such that the extraction of time series data features is more efficient.

For the input sequence $[s_1, \ldots, s_t], s_t \in \mathbb{R}^d$, the specific architecture is not only to maintain the aforementioned hidden layer state $h_t \in \mathbb{R}^H$, but to maintain the memory unit state $c_t \in \mathbb{R}^H$ as well.

$$\begin{aligned} i_t &= \sigma(U_i h_{t-1} + W_i s_t + b_i)), \ f_t = \sigma(U_f h_{t-1} + W_f s_t + b_f)) \\ o_t &= \sigma(U_o h_{t-1} + W_o s_t + b_o), \ \widetilde{c}_t = tanh(U_c h_{t-1} + W_c s_t + b_c)) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \widetilde{c}_t, \ h_t = o_t \odot tanh(c_t) \end{aligned} \tag{1}$$

where $U_i, U_f, U_o, U_c \in \mathbb{R}^{H\times H}, W_i, W_f, W_o, W_c \in \mathbb{R}^{H\times d}, b_i, b_f, b_o, b_c \in \mathbb{R}^H$ are model parameters. The specific LSTM unit structure is shown in Fig. 1.

Based on the LSTM model above, we design a deep learning framework for fault diagnosis, which is based on the multivariate time-series data. First, we split the whole time period to several segments for the preparation for our sequence model and transform it to the LSTM input vector. The specific function could be described as follows:

$$\kappa_{i_t} = \Phi(s_{mt:mt+m}) \tag{2}$$

where $\kappa_{i_t} \in \mathbb{R}^{mn}$ are the i_t-th network inputs, where m is the intercept window size and n is the dimension of this time series data. Φ is the preparation function, and s is a time-series signal segment $s \in \mathbb{R}^{l\times n}$ from n sensors and l is the series time length. The preparation function with many sensors can not only be a single cut-off function, but also including some dimension reduction operations like PCA, variation auto-encoder and so on, depending on different scenarios. With this function, the fault diagnosis LSTM network architecture is shown in Fig. 2. We first process the training data into the segment with

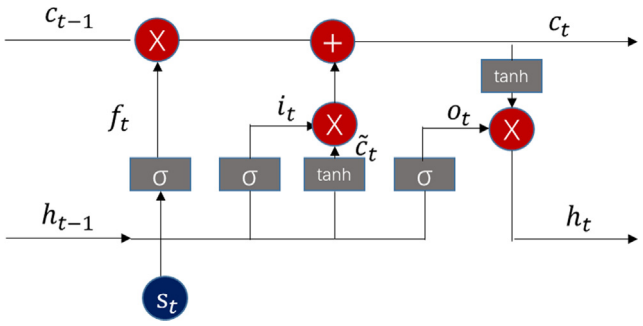


Fig. 1. Diagram of LSTM structure.

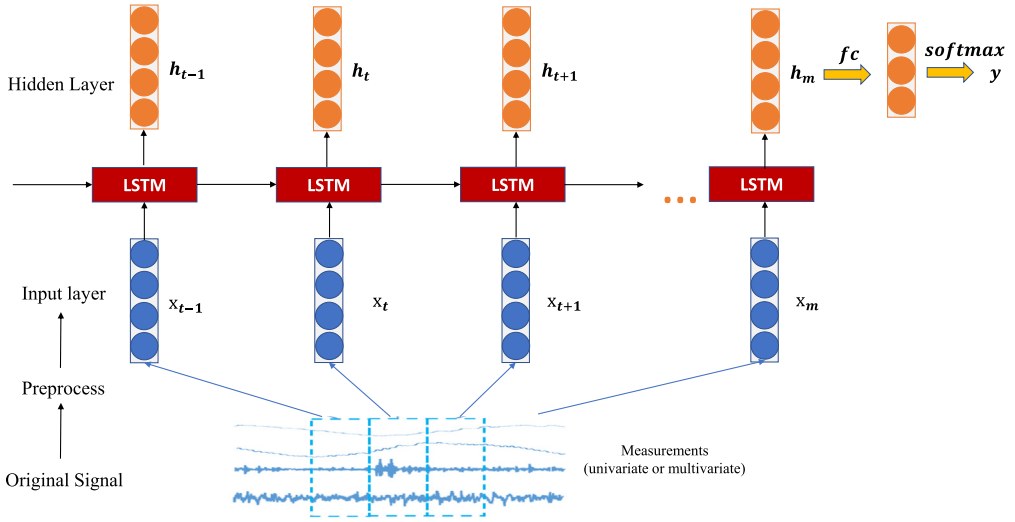


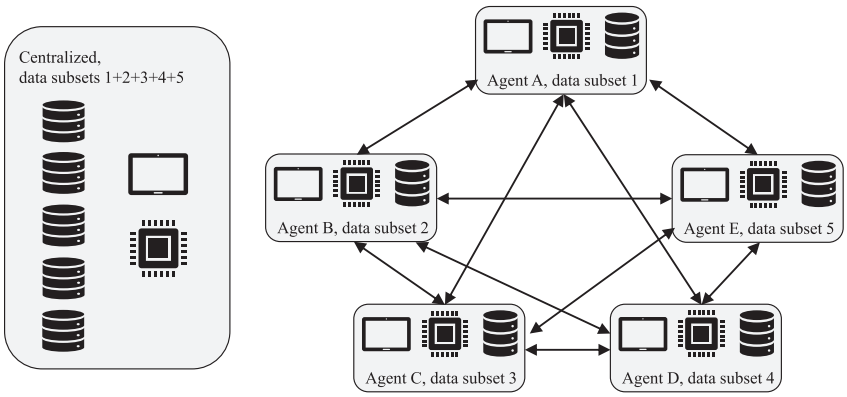
Fig. 2. Diagram of fault diagnosis LSTM networks.

length m. Then the input data for each time's LSTM is fed and we need to learn a suitable hidden layer representation h_t. With the final stage hidden layer, the classification task is finished with a fully-connection and softmax layer with a cross-entropy loss. With a sufficient number of input data and final labels, parameters for the model can be learned. For the future data, we also process it to the segment and identify the state in the period of segmentation. In the following part, for simplification, we denote the whole LSTM model as f, where $f : \mathbb{R}^{m\times d} \to \mathbb{R}$.

3. Collaborative deep learning architecture for distributed complex systems

3.1. Problem formulation

To formulate the collaborative learning problem properly in a distributed renewable energy system, we leverage the basis of the graph theory. Consider the distributed renewable energy system as a network system that involves N agents such that the underlying graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ signifies the set of agents and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ indicates the set of edges. Specifically, the schematic diagram of such a network system is shown in Fig. 3 (b). Each agent is equivalent to the aforementioned computing node, which actually can be one single computer or a cluster of computers placed in some computer centers. For any two agents i, j in the network system, if $(i, j) \in \mathcal{E}$, these two agents are able to communicate their own information with each other. We also define the neighborhood of any agent $i \in \mathcal{V}$ in this context. It is denoted by $Nb(i) := \{j \in \mathcal{V} : (i, j) \in \mathcal{E} \text{ or } j = i\}$. Throughout this paper we assume that $\mathcal{G}$ is *undirected* and *connected*. As discussed before, we are interested in the fault diagnosis problems in distributed complex systems where different fault data is stored in various computing nodes instead of the same node. Thus, we denote by $\mathcal{D}_i, i \in \mathcal{V}$ each training data set, which comprising q_i samples corresponding to the i-th agent and by $f^i(x)$ the local objective function to be minimized for each agent, where $x \in \mathbb{R}^d$ is the decision variable to be optimized by all agents (i.e. the parameters to be learned in all LSTM networks), $f^i : \mathbb{R}^{m\times d} \to \mathbb{R}$ is only known by agent i. It should be noted that LSTM networks are applied in this work for the objective func-



(a) Centralized learning (b) Distributed learning

Fig. 3. Distributed deep learning architecture.

tion, and the reason is explained in Section 2.2 that the LSTM networks are suitable for the fault diagnosis problem [13]. While other types of machine learning or deep learning models can also be applied as the objective function in the presented framework, and the selection of the objective function mainly depends on the problem to be solved, such as diagnosis, prognosis, or other tasks. We also define that $q = \sum_{i=1}^{N} q_i$ as the total number of samples.

The basic problem of collaborative deep learning is to solve the optimization problem to train the model in each computer to be as accurate as possible in the real test data. To achieve the goal of data privacy protection, the constraint that the amount of information communicated between the nodes is strictly less than the amount of information of the original data should be set in this problem.

More specially, in this problem true (raw operating) data could not be transferred but a more accurate model needs to be derived in each node than the model that is trained by only making use of local data. Mathematically, such an optimization problem is expressed by

$$\min \frac{1}{q}\sum_{i=1}^{N}\sum_{z\in\mathscr{D}_i} f_z^i(x). \tag{3}$$

where z represents a sample in $\mathscr{D}_i$. When solving the above optimization problem, each agent has its own estimation of x such that Eq. 3 can be rewritten as

$$\min \frac{1}{q}\sum_{i=1}^{N}\sum_{z\in\mathscr{D}_i} f_z^i(x^i),\ \text{s.t.}\ x^i = x^j,\ \text{if}\ (i,j)\in\mathscr{E}. \tag{4}$$

In this problem, as we mentioned before, the training model is a complex deep learning model, and the goal of it is the classification problem of fault categories. When applying the collaborative deep learning framework to our specific scenario, the difficulty of the problems has two folds: one is that the algorithm should take the network's topology and communication constraint into account and get the useful information transferred in the network; the other is that solving the Eq. 3 could be computationally intractable, as it is difficult to search the global optimal solution. Regarding the first difficulty, the proposed algorithm in this paper considers communicating their model weights among agents. A specific communication layer will be designed to help address such an issue. For the second difficulty, however, fortunately, in [30], the authors have shown the convergence to a feasible first-order stationary point even when f^i is highly non-convex. They also showed explicitly the relationship between the convergence and the network topology, which allows us to obtain the quantitative impact of the network topology in our case. Please refer to [30] for more details on the theoretical analysis.

3.2. Collaborative deep learning algorithm

To solve the above problem in Eq. 4, a collaborative deep learning algorithm is proposed. The basic rules of the algorithm follow the idea of gradient descent. First, the local agents utilize their LSTM models and parameters x^i to derive the prediction and then the loss. From the backpropagation, the gradients for each agent could be obtained as $g(x^i) = \nabla f^i(x^i)$ and the parameters x^i for each agent could be updated as follows:

$$x^i = x^i - \eta g(x^i), \tag{5}$$

where η represents the learning rate (step size). Next follows the key part of the collaborative learning. To derive other agent's information with privacy preservation, we need to conduct parameters' transferring, which is showed specifically as follows:

$$x^i = \sum_{j \in Nb(i)} \pi_{ij} x^j, \tag{6}$$

where π_{ij} denotes an element at the i-th row and j-th column of the weight matrix $\Pi \in \mathbb{R}^{N \times N}$. Throughout the paper, we assume that Π is *symmetric and doubly stochastic*. If the agent j is not adjacent to agent i, the weight π_{ij} could be set to 0. This algorithm could converge well with the theoretical proof in [30].

For visualization, Fig. 4(a) shows the process of the computation and communication based on the algorithm presented in [30]. Different from the centralized manner in 3(a), this distributed learning scheme in 3(b) trains the model by exchanging the parameters of the model, and the privacy is preserved by not transmitting the raw data. However, in the implementation of the collaborative learning for distributed renewable energy systems, a few adjustments need to be taken. As the algorithm is sequentially updated in [30], the first adjustment in this context should be done for the parallelism. In addition, due to the complexity in most deep neural networks, the Adam optimizer [31] has been shown empirically better than the SGD. Thus, we also provide an variant of the distributed learning algorithm using the Adam optimizer. (see Fig. 5).

Parallelism The structure for parallelism is shown in Fig. 4. In Fig. 4(a), the parameter transferring from Agent i to j (represented by the green line) occurs after the update of Agent i, which is a sequential updating strategy and it becomes the bottleneck of the algorithm and slows down the convergence rate. In this work, a real synchronous collaboration is adopted, as shown in Fig. 4(b). A communication layer for the synchronization, is responsible for collecting the parameters from all of the neighbors (defined by the graph) and then it triggers the next step. With this structure, all the agents are able to update their parameters synchronously within the same step. The problem that whether there exists a agent that might fall behind and become a cumbrance to the whole distributed learning system is naturally solved.

Synchronous-CDAdam As shown in [31], the Adam optimizer has been adopted to many deep learning optimization problems and resulted in a better performance compared with simple SGD. Hence, in order to better solve our problem, we extend the framework of the CDSGD in [30] to the Adam optimizer. We take advantage of the averaging for consensus and apply this perspective to the Adam optimizer. Adam adopts the momentum and adaptive learning rate to control the learning process. We also add the moment to the shift of the local parameters, which preserves the tendency of the local shift to reduce the variance caused by the stochasticity of data. We next change the learning rate since all of our agents have been synchronized and the local learning rate adjustment could be the same and conducted locally. Therefore, the CDAdam optimizer is formed in this work where a consensus and averaging substep is added after the updates in the original Adam algorithm for each local agent because the two main steps of Adam can be both conducted locally with the analysis above. The specific algorithm is shown in the Algorithm 1. We make use of the local gradients and the stored momentum to construct the momentum locally as follows:

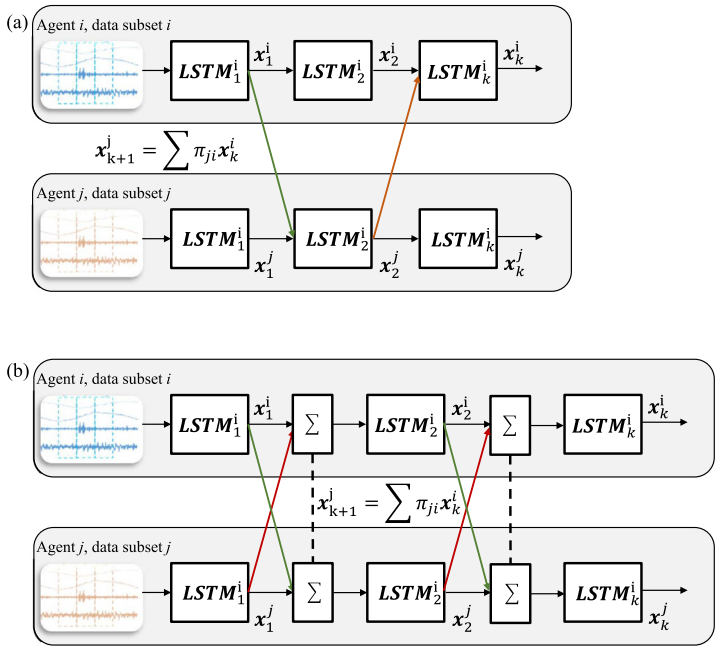


Fig. 4. Communication layer design.

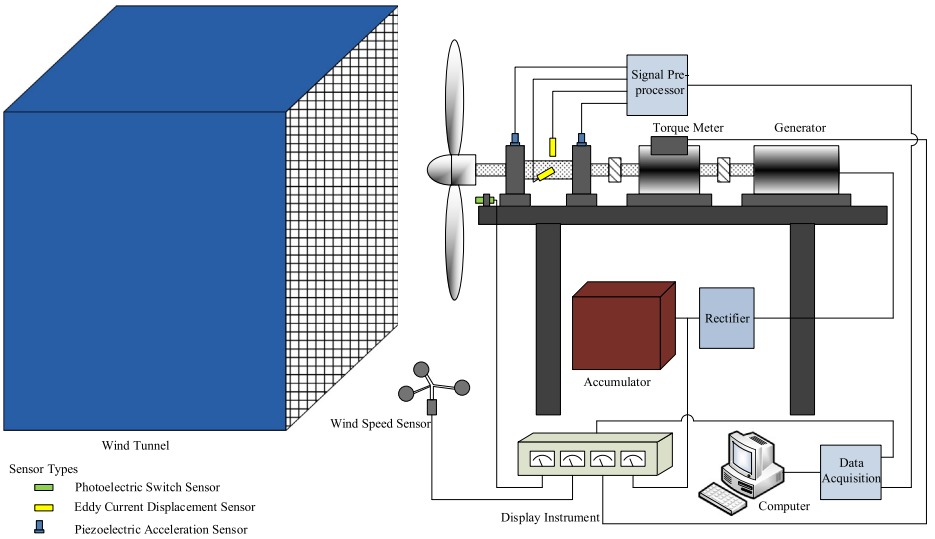


Fig. 5. Test rig of Direct Drive Wind Turbine Fault Simulation System.

$$m_{k+1}^i = \beta_1 \cdot m_k^i + (1-\beta_1)\cdot g_k(x_k^i), \tag{7}$$

where m_{k+1}^i means the local momentum and β_1 denotes the ratios of the step influenced by the momentum. After the broadcast and consensus, this item could become the momentum for the virtual model who owns data of all agents to conduct training centrally. Then the adaptive learning rate steps are as follows:

$$V_{k+1}^i = \beta_2 \cdot V_k^i + (1-\beta_2)\cdot g_k(x_k^i)\cdot g_k(x_k^i), \tag{8}$$

where V_{k+1}^i is used for local learning rate adjustments. For this adaptive learning rate changing, since we conduct synchronization and the agents' consensus, this learning rate adjustment cannot vary much for all the agents and we achieve the results of adaptive training for the total distributed system. Finally, we conduct the averaging between all the agents' parameters to make consensus through our communication layer. The empirical results demonstrate that this Synchronous-CDAdam algorithm can speed up the convergence in our experiments without losing the characteristics of our distributed learning framework.

Algorithm . Synchronous-CDAdam (for node i)

Input: learning rate η, parameters weight matrix for node i π_i, the number of epochs M, constants β_1, $\beta_2 \in [0,1)$, $\epsilon = 10^{-8}$

Output: the models' parameters in node i x_M^i

Initalize all the nodes' parameters x_0^i, exponential moving average of gradient values m_0, exponential moving average of squared gradient values V_0, $k=0$

while *epoch* $k < M$ **do**

if calculation for all nodes not done **then**

wait and set the signal of completion

end if

broadcast x_k^i to and receive x_k^j from neighbors

$m_{k+1} = \beta_1 \cdot m_k + (1-\beta_1)\cdot g_k(x_k^i)$

$V_{k+1} = \beta_2 \cdot V_k + (1-\beta_2)\cdot g_k(x_k^i)\cdot g_k(x_k^i)$

update: $p_k^i = x_k^i - \eta \frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k}\frac{m_{k+1}}{\sqrt{V_{k+1}+\epsilon}}$

consensus: $x_{k+1}^i = \pi_{ii}p_k^i + \sum_{j\in Nb(i), j\neq i}\pi_{ij}x_k^j$

$k = k+1$

end while

Communication layer design To realize the parallelism in the collaborative learning framework, we add a communication layer for synchronization. In this part, we present the effect of this block in detail. To realize the aforementioned parallelism, we need an extra controller to guide the local agent and help synchronization. The communication layer is designed for this. To achieve the consensus step, we need this block to collect the parameters and do averaging from the neighbors. The agent does not start because it does not know whether synchronization has been achieved. Therefore, the block sends the signal to broadcast to other agents that this agent has finished the task of its iteration and waits for other agents' signal. After receiving all the signals, it will start the next step. Aided by this communication layer, our architecture becomes more clear and each mode is taken apart for easier design or direct applying mature software, which reduces the burden for the system's implementation. In addition to that, more functionalities could be developed by considering time delay, topology etc., which would make our algorithm closer to reality.

4. Results and discussions

The case studies for implementing the presented collaborative deep learning framework consist of three scenarios, (i) the data assigned to the agents is with i.i.d. assumption (i.e., the distributions of the training data in terms of the inputs and outputs are same), two case studies using wind turbine test rig [32] and FDI [33] data sets are applied in this scenario (noted as `Scenario I`), (ii) the data in the agents is imbalanced, the wind turbine data set is applied in this case by assigning imbalanced data to two or more agents (noted as `Scenario II`), and (iii) the data in the agents is with different distributions (noted as `Scenario III`), bearing data set by Paderborn University [34] is applied in this case, where each agent is assigned with healthy and fault data from different bearings. We use these tasks to check the correctness, robustness against data distribution and generalization in the real world cases.

4.1. Data sets

The used data sets are explained in this section, and the four case studies in the aforementioned three scenarios will be illustrated in the following sections.

Wind turbine data set. A direct-drive wind turbine test rig in the laboratory [32] is applied and the data in the normal condition and 11 fault conditions are collected.

The setup of the test rig can refer to [32]. The whole system obtains the wind through a wind tunnel using an axial flow fan. The fan speed is adjusted by changing the wind speed such that different working conditions could be simulated. Our main characteristic data is derived from the time series data collected by the generator's horizontal and vertical direction displacement sensors, and front and rear acceleration sensors for bearing supports.

In the experiment, the sampling rate is 20 kHz. Twelve machine conditions are simulated in test rig including 11 fault conditions and one normal condition, as listed in Table 1. All these faults can fundamentally cover the common failure modes from wind wheel to drive-chain [4,12].

Data preparation of the wind turbine data sets will be introduced in `Scenario I` and `Scenario II` of the following sections, with different settings of the data assignment to the distributed agents.

Fault detection and isolation (FDI) data set [33]. The fault detection and isolation (FDI) is based on a 5 MW wind turbine model proposed in [33]. The settings of the fault conditions used in this work follow the instructions in [35]. The measurements and data generation process follow the method described in [13]. FDI data set is applied in `Scenario I`) where the training data is equally assigned to each distributed agent. The training data consists of 8 measured parameters and 8 conditions, which is showed in Table 2.

The measurements used in this work are listed in Table 3. Other data generation process follows the method described in [13].

Table 1
Tested conditions (normal and 11 faults) on wind turbine test rig.

Condition ID	Description
0	Normal
1	Misalignment, horizontal direction
2	Misalignment, vertical direction
3	Loosening, front bearing support
4	Loosening, back bearing support
5	Rolling element fault, front bearing
6	Inner-ring fault, front bearing
7	Outer-ring fault, front bearing
8	Mass unbalance of wind wheel
9	Variation in airfoil of blades
10	Yaw fault
11	Aero-asymmetry of wind wheel

Table 2
Description of FDI dataset.

ID	Description[35]
0	Normal condition
1	Pitch actuator, change in dynamics: high air content in oil ($w_n = 5.73rad/s, \xi = 0.45$)
2	Pitch actuator, change in dynamics: pump wear ($w_n = 7.27rad/s, \xi = 0.75$)
3	Pitch actuator, change in dynamics: hydraulic leakage ($w_n = 3.42rad/s, \xi = 0.9$)
4	Generator speed sensor, scaling (gain factor equal to 1.2)
5	Pitch angle sensor, stuck (fixed value equal to 5 deg)
6	Pitch angle sensor, stuck (fixed value equal to 10 deg)
7	Torque actuator, offset (offset value equal to 2000 Nm)

Table 3
Description of FDI dataset.

ID	Measurement[36,33]
1	Wind speed
2	Rotating speed of wind wheel, w_{rm1}
3	Rotating speed of wind wheel, w_{rm2}
4	Rotating speed of generator rotor, w_{gm1}
5	Torque of generator, τ_{gm1}
6	Pitch angle of blade 1, β_{1m1}
7	Pitch angle of blade 2, β_{2m1}
8	Pitch angle of blade 3, β_{3m1}

Bearing data set by Paderborn University[34]. This data set is obtained by the modular test rig at Paderborn University, where the details are described in [34]. To form the distributed learning setup for `Scenario III` with five agents, the normal and two fault conditions (inner-ring fault and outer-ring fault) are chosen from different bearings and each agent in the distributed network is assigned with unique bearing data. In this context, the training data for the agents is non-iid, and the distributions are significantly different. We will detailed discuss about the fault type of each agent in the following part of `Scenario III`.

4.2. Case studies on three scenarios

Distributed agents (complex systems) setting. Five agents are assumed in the distributed network for the case studies in this work, where each agent (node) represents a local data center that holds data collected from the local complex systems. Note that the number of agents can be different in different applications.

Distributed network's topology and communication setting. The distributed network is represented by a fully-connected graph, which means that the communication occurs between each two agents. For each agent, a local deep learning model can be formed with local data and the parameters of the model will be exchanged following the setting of the distributed network.

Comparison methods. Besides the distributed learning framework, the central and local schemes will be compared in the following case studies (as listed in Table 4). For the **Central training** strategy, data needs to be transmitted to a central data center (which is probably not idealistic in reality). As all data can be seen by the deep learning model in this setup, the performance will be better and this can be treated as the upper bound for the distributed learning strategy (i.e., collaborative deep learning model in this work). For the **Local training** strategy, the deep learning model is only trained by the data in each agent. The local data would be only input to the local agent. The final one is our **Distributed training**, which allows the parameters' communication based on the local strategy.

Table 4
Three training strategies: central, local and distributed training.

Strategy	Implementation
Central	Training in the central data center, where fault data is transmitted to the center from local agents.
Local	Training separately in each local agent (node), without communication among the distributed complex systems.
Distributed*	Training in each agent with communication of model parameters among the distributed complex systems (distributed network), while raw data is not transmitted.

* Collaborative learning proposed in this work.

With this setup, case studies on the aforementioned scenarios are carried out. Note that the improvement of performance on the presented collaborative deep learning algorithm over the local training model can be treated as the gain of the collaborative deep learning framework.

4.2.1. `Scenario I`: collaborative deep learning with balanced data sets

Wind turbine data set and FDI data set are applied in this scenario, noted as *case i* and *case ii*.

Case i: wind turbine data set with balanced setting

Data preparation and hyper-parameters. The measurements are collected from four sensors in the test rig and experiments on single sensor and multiple sensors are implemented. The input length of the time-series data is 4000 for each sensor and reshaped into $200 \times (n \times 20)$, where n is the number of sensors, 20 is the window size and 200 is input sequence length. $n = 1$, for single-sensor case, and $n = 4$, for multi-sensor case. A two-layer LSTM structure is applied where 200 hidden units are included in each layer. The total number of samples for each condition is 300. CDAdam is applied with the learning rate 0.01. The hyper-parameter search and fine tuning process follow the principles in [13].

Performance evaluation. The training and testing accuracy for single-sensor and multi-sensors are shown in Fig. 6. It shows good convergence rate presented by the collaborative learning algorithm, indicating that the distributed algorithm and the communication of model parameters do not affect the convergence of model and we would derive a stable solution through this training process. Also, the number of epochs (about 1,000 for multi-sensors and 2500 for Sensor 1,2) needed to the convergence in the collaborative learning is larger than that in the central learning scheme (as shown in [13]), which is a drawback for the distributed learning algorithm that the convergence rate is smaller. (see Fig. 7).

Comparisons. Comparing the accuracy of the test set of this distributed model with the 20% data local training model without communication in network, in all cases, the minimum value of each node of this distributed model is greater than that above 20% data training results. It shows that at each node, better results are obtained than the local strategy's result, indicating that such a distributed deep learning algorithm can improve the accuracy of each node model, thus achieving more accurate faults diagnosis. Later, the bar graph also compares the distributed and fully-center algorithm's performance which is an upper limit for the distributed model. As can be seen from the figure, this model has been close to the upper limit in multi-sensors, in which the gap is only 0.3%. For some single sensor data experiments, the distributed deep learning fault diagnosis model can guarantee data security and take full advantage of fault data to derive a result as accurate as the central strategy with raw data collection. In other words, each node with fault data participating in the distributed training process can obtain a more accurate fault diagnosis deep learning model.

In terms of the classification performance on each condition, the confusion matrix Fig. 8 shows that mis-classification rates for 300 samples each condition. In the figures, we show the percentage for the confusion of the fault type.

From the figure, the error of condition 7 is significantly reduced for the collaborative learning model. For condition 10, the local learning model almost misclassifies all of them, while the collaborative learning model can classify around 10% of them correctly. From the t-sne of the hidden states plots in Fig. 9, we could find out that the condition 7 and 10 could be hard to distinguish from condition 9 and 11. Therefore, more information would be needed for a better performance of the classifier. Actually our model takes advantage of all the parameters and gathers more information. It should work better in condition's recognition, which is consistent with our confusion matrix.

Case ii: FDI data set with balanced setting

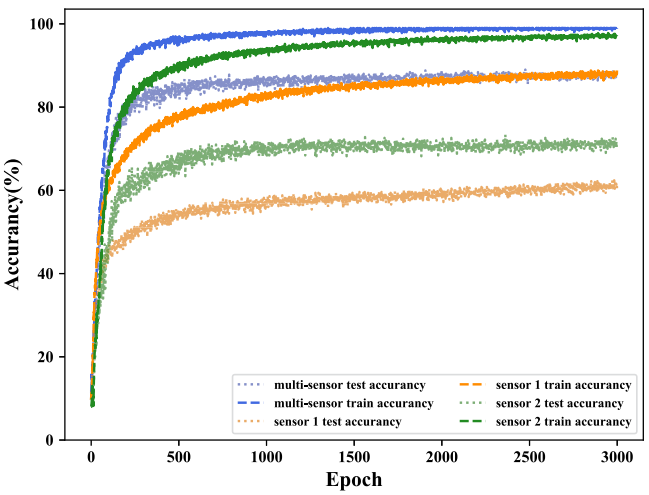


Fig. 6. Training and testing accuracy of Case i for collaborative deep learning framework.

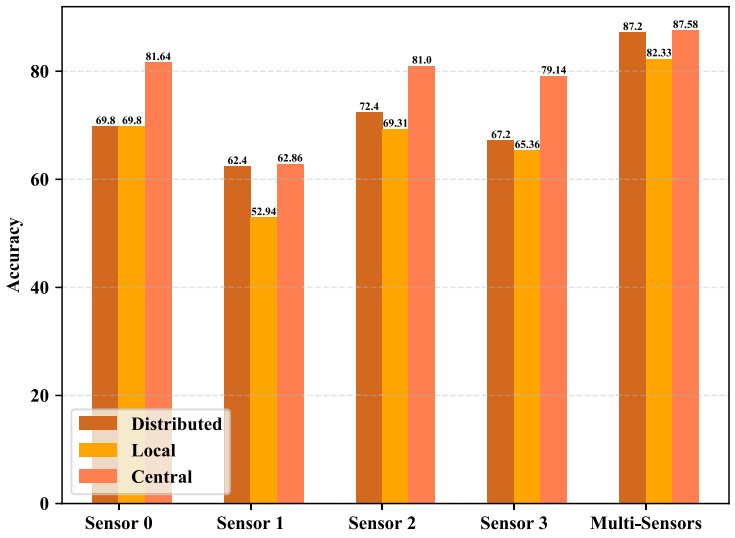


Fig. 7. Performance on three learning strategies with single-sensor and multi-sensor situations in Case i.

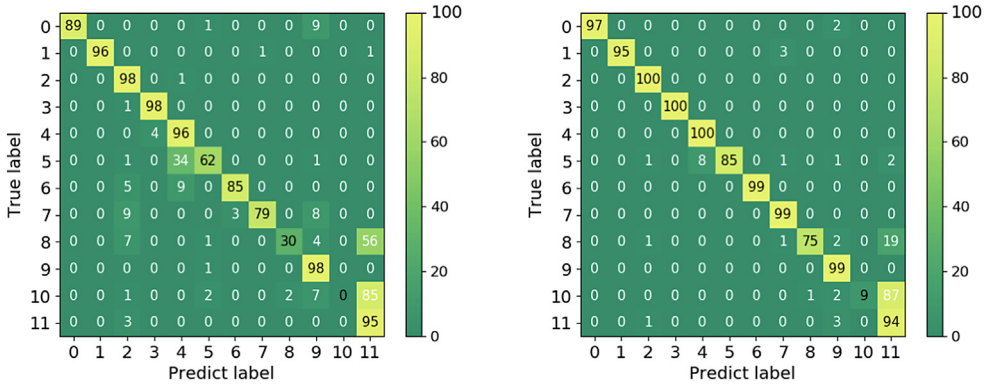


Fig. 8. Confusion matrix with local learning model and collaborative learning model for Case i.

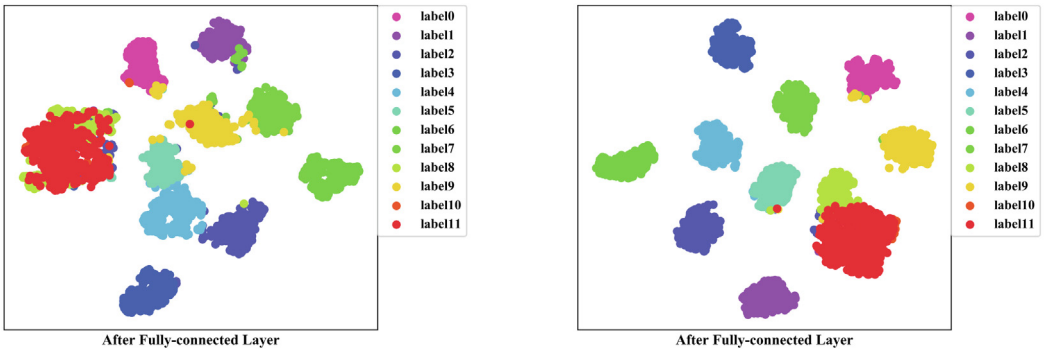


Fig. 9. T-sne visualization with local learning model and collaborative learning model in Case i.

Data preparation and hyper-parameters. The deep structure used in this case is the same as that in case i. The input dimension is 10 for the one-layer LSTM model, where the number of hidden units is 140. CDAdam is applied with the learning rate searched from 0.0001 to 0.01. More detailed process for the data could be showed in [13].

Performance evaluation and comparison. The training process is shown in Fig. 10(a), where the collaborative learning model converges after around 1800 epochs and the testing accuracy approaches to 100%. It reflects that our model could be generalized to other dataset.

Fig. 10(b) shows that the performance of collaborative learning model is close to central learning model (98.9% vs 99.4%), which indicates that for other dataset, the collaborative one can also achieve a good performance without the need of data transmission (which the central learning model needs). Compared with the local learning model (the best performance is shown in the figure), the accuracy of the collaborative learning model is over 14% higher, which implies the necessity of a distributed learning framework for the exchange of the classification knowledge represented by the data in other agents. In brief, this experiment verifies that our proposed distributed framework could be generalized to other datasets.

From the above two case studies, the collaborative deep learning framework successfully extracts the information represented by the data from other agents to improve the performance. In the mean time, since no data transmission during the collaborative learning process, the data privacy is preserved.

4.2.2. `Scenario II`: collaborative deep learning with imbalanced data

With the assumption that fault occurrence depends on the operating and environmental conditions, imbalanced setting is often observed in reality, especially for the scenario that the data is collected from distributed complex systems.

Case iii: Wind turbine data set with imbalanced setting

Imbalanced data and model setting. Based on the data preparation in case i with five agents, we change the distribution of the agent. In case i, each agent has the equal data of all the conditions. The data ratio for the conditions 4 and 5 is assigned as $1:1:1.2:0.8:1$ and the data ratio for the conditions 7 and 8 is assigned as $1:1:0.8:1.2:1$. The specific meaning of each condition could be viewed in Table 1. It could be found out that agent 3 and 4 have different distributions from other agents.

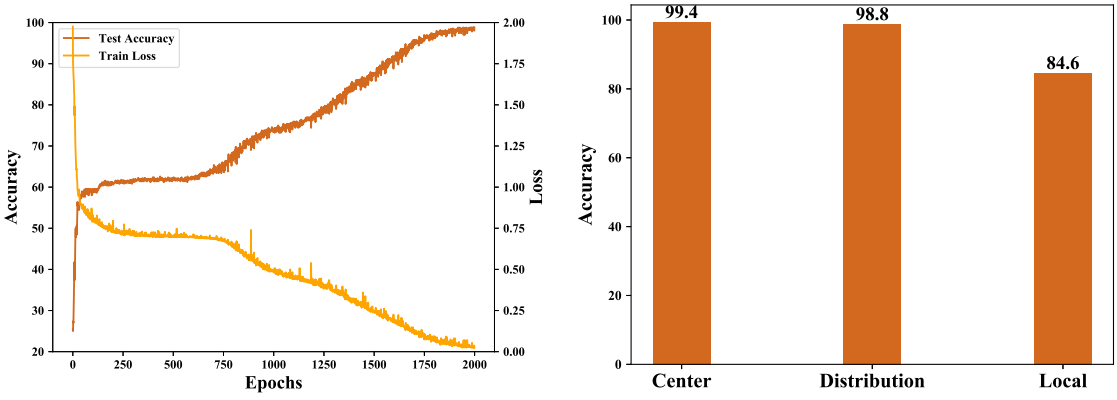
LSTM is also applied in this case with the same setting in case i in terms of the model structure, the learning rate, optimizer selection and hyper-parameter searching.

Performance evaluation.

First we still analyze the convergence and average performance comparison as before in Fig. 11. Fig. 11(a) shows that our algorithm converges well for this imbalanced data set after about 1800 epochs. Actually, we only need 500 epochs to get a model with great performance in this case. It means that our distributed model could be also stable in the data imbalanced case. Compared with the center method in Fig. 11(b), our performance approaches the upper bound and we also gain a performance boost by around 4% compared with the local non-distributed algorithm. This part verifies that our distributed model could be extended to imbalanced dataset, which is more closer to reality.

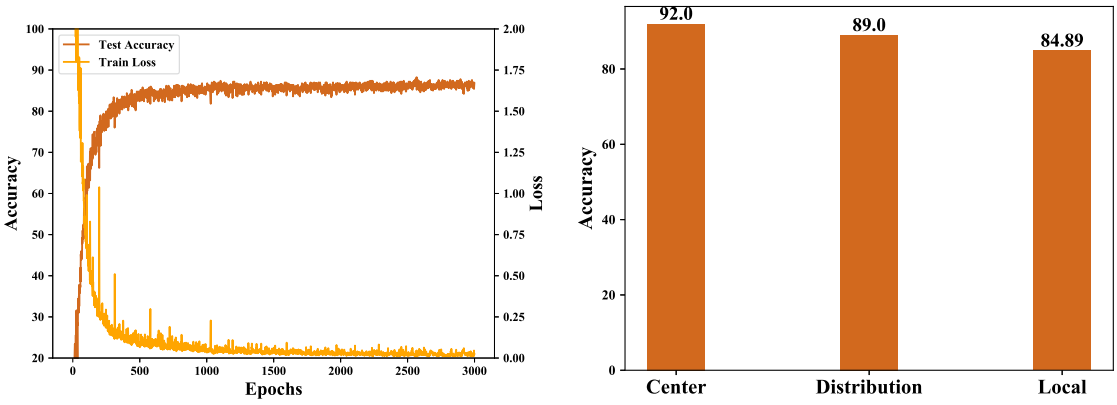
Then we turn our attention to the two agents with imbalanced data (agents 3 and 4). Fig. 12 shows that for the local strategy, the accuracy presents a drop on the agents with imbalanced data (1.2% decrease in agent 3 compared to agent 1, and 3.6% drop in agent 4), this is probably caused by the training difficulty for the imbalanced data in the two agents and they lose some information respectively. However, the collaborative learning model achieves close or even better performance on the two agents (1% increase in agent 3 compared to agent 1, and 0.1% drop in agent 4) because they could find these information lost back from other agents.

This is also reflected in the misclassification rates for agent 4 who has less data of conditions 7 and 8. As shown in Fig. 13 for the confusion percentage, the misclassification rates of conditions 7 and 8 are significantly decreased in the collaborative learning model, which shows our model could really help local agent to make up their shortcomings.



(a) Convergence of collaborative model (b) Performance comparison

Fig. 10. Convergence for collaborative model and performance comparisons with central and local learning strategies in case ii.



(a) Convergence of collaborative model (b) Performance comparison

Fig. 11. Convergence and performance comparison for imbalanced wind turbine data (case iii).

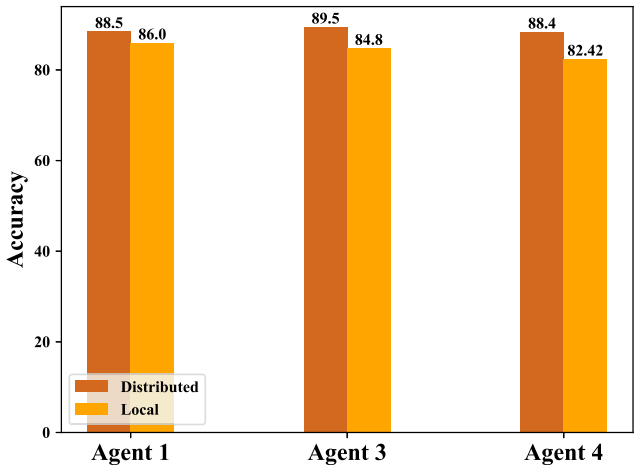
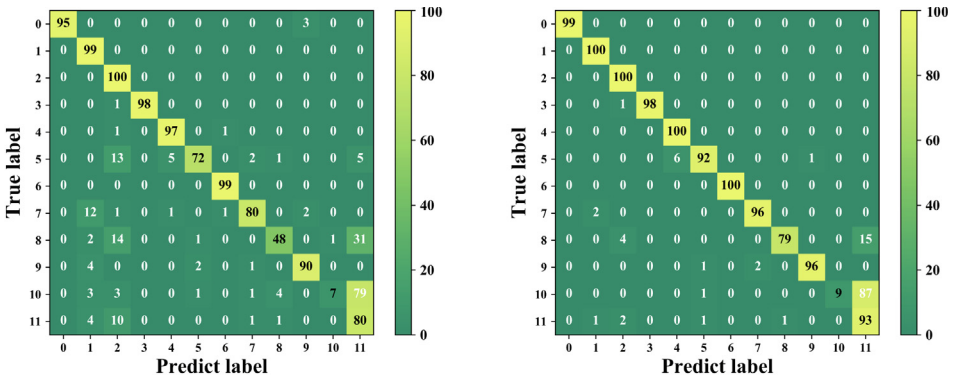


Fig. 12. Comparison between distributed and local algorithms for different agents (case iii).



(a) Local model (b) Collaborative model

Fig. 13. Confusion matrices for local and collaborative learning algorithms (case iii).

This case shows that collaborative deep learning algorithm could boost the agent with more capacity in classifying the conditions with imbalanced training data.

4.2.3. Scenario III: collaborative deep learning with different distributions

Through the above case studies, the collaborative deep learning shows its efficacy in improving the performance by exchanging the model parameters with no need of raw data transmission. A more realistic scenario discussed here is that, the distribution of data in different agents may vary a lot, as different mechanical or energy systems are not consistent in terms of the degradation process of the components in the systems (as listed in Table 5, where the components are naturally different). With the assumption that fault occurrence depends on the operating and environmental conditions, imbalanced setting is often observed in reality, especially for the scenario that the data is collected from distributed complex systems in this work.

Case iv: Bearing data set with different distributions

Data preparation. The experimental data from 15 bearings [34] is chosen including normal, inner-ring fault and outer-ring fault conditions, where each condition consists of data from five bearings. To form the case study with different distributions of data, we assign each agent with the data of three bearings (one bearing for each condition), as listed in Table 5. In this context, each agent of the distributed mechanical systems consists of three bearings that are different from other agents, and the distributions of the input data are completely different. We choose the data of 5 bearings and 3 states of these bearings including normal, outer ring fault and inner ring fault. The specific bearing codes are shown in Table 5.

Model structure and hyper-parameters. For the acceleration measurement, 5000 data points are selected for a training sample and reshaped into 200×25 for the LSTM inputs after searching the first dimension from 50 to 400. The first dimension represents the sequence length for LSTM and the second denotes the dimension for each LTSM input. A one-layer LSTM network is built with 30 hidden units. Each agent is with 2400 training samples for the three conditions. CDAdam is applied and the learning rate is set as 0.01 after searching from 0.001 to 0.1. The typology is also a fully-connected framework with five local agents.

Performance evaluation. Fig. 14(a) shows the convergence process of the collaborative deep learning algorithm, which presents obvious variance as of the parameter exchange during the communication from other agents with different data distributions. Although the collaborative learning process needs more time to converge, which is actually about 2,500 epochs, the benefit is also significant as shown in Fig. 14(b), where the accuracy for the collaborative learning model is 92.4% with over 31% increase than the local learning model.

The confusion matrices of all agents are shown in Fig. 15. In the figures, we show the percentage for the confusion of the fault type with 800 samples in each type. From the figures, we could view that the collaborative learning method (in the top panel) classifies the three conditions in most cases for all agents, while the local learning method (in the bottom panel) fails in lots of cases. The collaborative learning method outperforms in the situations that the individual agent can diagnose the unseen mechanical systems from other agents, which indicates a good generalization capacity.

In case iii and case iv, we test the generalization of our algorithm and push it more to realistic scenarios. The presented collaborative deep learning algorithm could also be adaptive to the imbalanced data and non-iid data with different distributions. It means that a collaborative system could be established between the agents with different types of fault data, which could make the present framework applicable in practical applications.

4.3. Discussions

Data representation and privacy preserving. The presented collaborative deep learning framework is able to extract the useful information for the classification task from other agents. As discussed in Section III, the raw data is not transmitted between the distributed network formed by distributed complex systems. Instead, the information (knowledge) for the classification is represented by the model parameters in deep learning structures. Thus the presented collaborative deep learning framework is able to capture the features of the data without the need of data transmission, which is a naturally privacy preserving method.

Performance. The collaborative deep learning framework outperforms the local learning scheme, which shows the necessity and advantages of the presented framework. Specifically, in case iv with a more realistic application scenario, the accuracy of the presented algorithm is over 31% higher than the local learning algorithm. With no need of raw data transmission,

Table 5
Data setting for the distributed mechanical systems.

System	Nor	OR	Damage	IR	Damage
1	K001	KA07	Drilling Holes	KI03	Electric Engraving
2	K002	KA22	Pitting	KI05	Electric Engraving
3	K003	KA03	Electric Engraving	KI017	Pitting
4	K005	KA06	Electric Engraving	KI07	Electric Engraving
5	K006	KA16	Pitting	KI04	Pitting

Nor: normal, OR: outer-ring fault, IR: inner-ring fault.

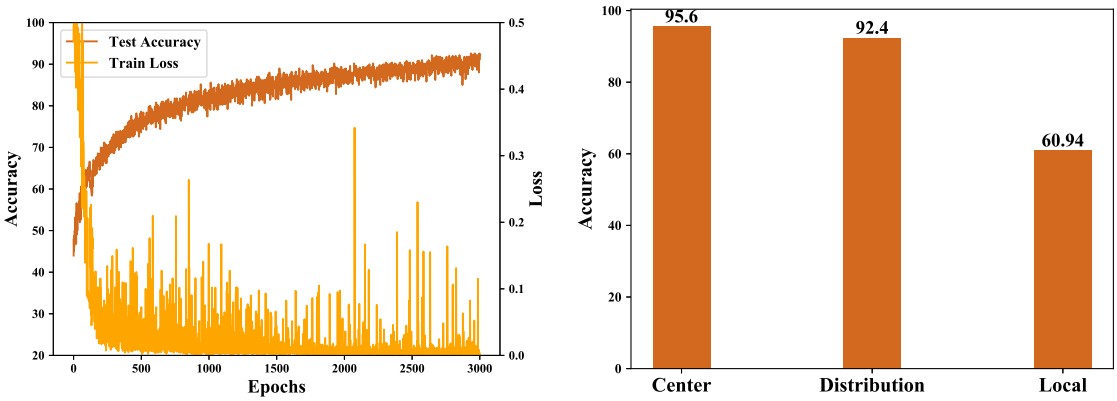


Fig. 14. Convergence and performance comparison in Case iv for training data with different distributions (case iv).

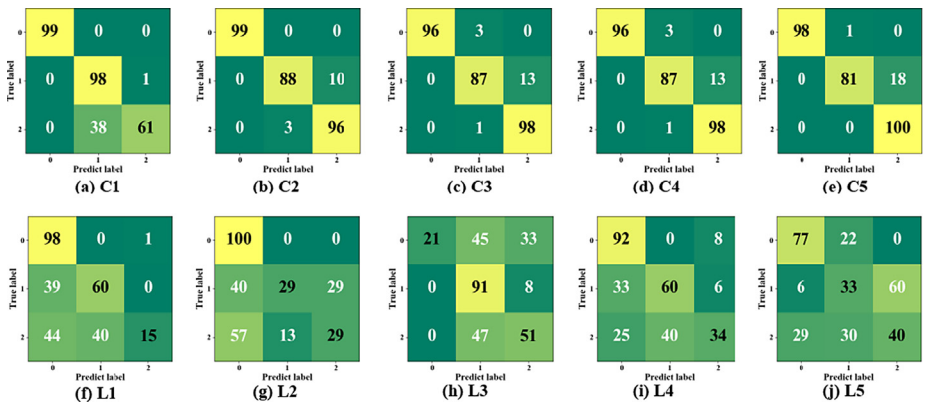


Fig. 15. Confusion matrices for Case iv for collaborative learning and local learning models. C1-C5: five agents trained by collaborated learning framework; 'L1'-'L5': five local trained agents.

the collaborative learning method is almost as accurate as the central learning method (which needs data transmission to the central server), which also shows the efficacy of the presented framework. Four case studies in diverse scenarios and applications verify its robustness.

Communication costs. Based on case i, the communication costs are estimated. The number of parameters in the LSTM model is 0.05 M, and the training process takes 1,000 epochs to the convergence. For our collaborative deep learning algorithm, the total amount of information transmitted is about 3.72 GB. For the central learning algorithm, the number of samples in this experiment is about 80,000 (4-dimensional tensor per sample). Thus the central learning algorithm requires 5.96 GB of raw data transmission. It implies that the collaborative learning reduces the communication costs. Furthermore, as the amount of data increases, the communication costs increase linearly for the central learning, while the collaborative learning does not increase if the model size does not increase. In this context, the collaborative deep learning framework will be more superior for the big data applications.

Load balancing and stability. Regarding load balancing, distributed computing makes full use of the distributed computing power, and minimizes the loads of the computing units distributed in different locations. With the collaborative deep learning framework presented in this work, the actual occupancy rate of each GPU is between 10% and 20%, and the central learning algorithm, the computing resources are occupied around 90–100%. From this perspective, distributed computing reduces the loading on each unit and improves the stability of the whole system.

Overfitting. With the transmitted parameters of the deep learning model, the collaborative deep learning framework avoids the overfitting problem when there is limited data in a local data center. For the fault diagnosis of complex (mechanical or energy) systems, the fault data is usually limited and is not willing to be shared in terms of the data privacy while quite a number of data is needed to train deep learning frameworks for better performance. In this context, the presented method provides a solution for deploying deep learning methods in the applications.

With shallow/deep models. Although we use LSTM in this paper, other deep structures or shallow machine learning methods can also be applied with this framework. Also, the topology of the distributed network could influence the performance. These extensions could be better discussed for future work.

4.4. Field implementation

For the real implementation, the flowchart is shown in Fig. 16. For the local agents, the architecture in Fig. 17 of the local application is designed based on the existing network applications in terms of the data storage, data loading, computing platform, etc. For the parameters sharing of the collaborative learning model, the 'signal' for each local agent can be defined as an iteration step, where the signal is broadcast once an iteration step is finished (or complete) in a local agent, and the parameters of the collaborative learning model are only updated when the signal is received from each of the local agent.

More specifically, the framework first needs the agents that participate in the training to sign a protocol to affirm providing the parameters of the collaborative model. During initialization, the local agents initialize their local models and then they broadcast packages to identify the topology of the training network and get the neighbors for communication. Then the training can be implemented by using either Pytorch or Tensorflow. After training, the agents use FTP protocol softwares to transfer the local parameters to other agents. They also broadcast their "complete" signal to all the agents in the network. Then if each agent receives all "complete" signals and all neighbor parameters, it will conduct the averaging and start for next epoch training. Finally, the local agent stops the training when the convergence or stopping criterion is satisfied.

Specifically, for the communication of the parameters of the collaborative learning model, the architecture is shown in Fig. 17. The data can be obtained from SCADA system or other data acquisition software, and database is usually used for storage. SQL tools can be applied to store and manage the local data. When training, we make use of the FPGA or GPUs or other hardware for computing. FPGA or GPU communicates with their local PCs with the model parameters through the existing training software like Pytorch or Tensorflow and local PCs transfer their data through a specific port, which is ruled in the protocol. The transferring is based on the simple data transferring protocol like FTP and the network layer protocol in this context is TCP/IP. Subsequently the other agents receive the packages and further change the parameters with their own hardware for training. In the practical implementation, there can also be sublayers like the encryption protocol to further guarantee the privacy. However, it is beyond the scope of this work, and will be considered in future work.

5. Conclusion

To address the data sharing and apply deep learning structures in distributed complex systems, this work presents a collaborative deep learning framework for fault diagnosis. By utilizing the scattered data in distributed mechanical or energy systems, the proposed framework forms a secure communication strategy with no need of transmitting the raw data, and obtains a consensus for distributed deep learning models which can be geographically located in different data centers. The results on four case studies validate the efficacy of the presented framework. It shows that our proposed framework

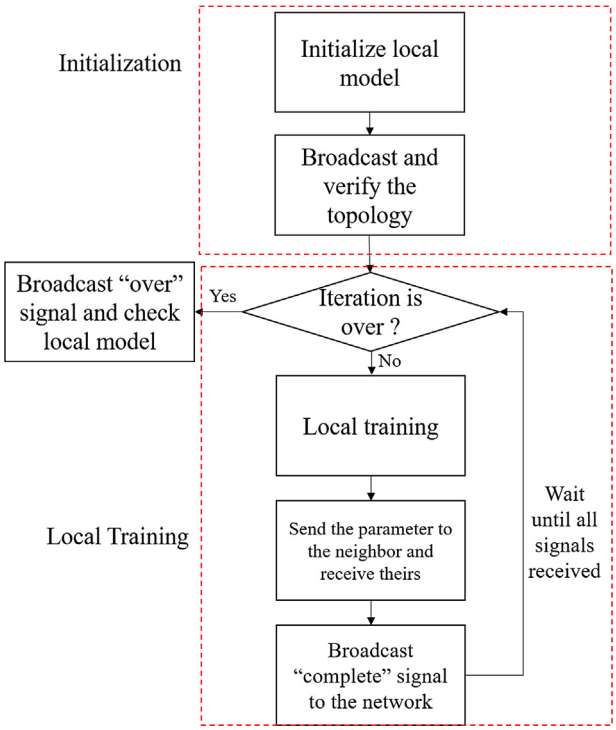


Fig. 16. The flowchart for implementing the proposed framework.

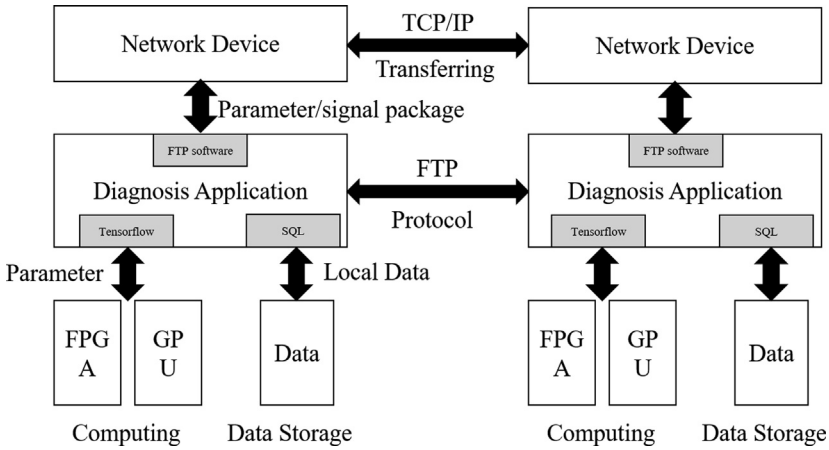


Fig. 17. The architecture for implementation of our framework.

is able to improve the diagnosis accuracy compared with the local learning models, and reaches close performance with central learning models which has high costs of data transmission. Also, the proposed framework is shown to be robust and adaptive for diagnosis problems with data that is imbalanced or from different distributions.

Future work will pursue, i) collaborative learning with more structures, ii) topology optimization constrained by communication costs of geographically distributed systems.

CRediT authorship contribution statement

Haoxiang Wang: Conceptualization, Methodology, Software, Investigation, Writing - original draft. **Chao Liu:** Conceptualization, Methodology, Data curation, Validation, Writing - review & editing. **Dongxiang Jiang:** Supervision, Resources, Funding acquisition. **Zhanhong Jiang:** Validation, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] K. Zhou, S. Yang, Z. Shao, Energy internet: the business perspective, Appl. Energy 178 (2016) 212–222.
[2] M. Zhang, J. Chen, S. He, L. Yang, X. Gong, J. Zhang, Privacy-preserving database assisted spectrum access for industrial internet of things: a distributed learning approach, IEEE Trans. Industr. Electron. 67 (2019) 7094–7103.
[3] S. Sun, F. Liu, S. Xue, M. Zeng, F. Zeng, Review on wind power development in china: current situation and improvement strategies to realize future development, Renew. Sustain. Energy Rev. 45 (2015) 589–599.
[4] W. Yang, C. Liu, D. Jiang, An unsupervised spatiotemporal graphical modeling approach for wind turbine condition monitoring, Renewable Energy 127 (2018) 230–241.
[5] R. Iqbal, T. Maniak, F. Doctor, C. Karyotis, Fault detection and isolation in industrial processes using deep learning approaches, IEEE Trans. Industr. Inf. 15 (2019) 3077–3084.
[6] T. Pan, J. Chen, Z. Zhou, C. Wang, S. He, A novel deep learning network via multiscale inner product with locally connected feature extraction for intelligent fault detection, IEEE Trans. Industr. Inf. 15 (2019) 5119–5128.
[7] M. Zhao, S. Zhong, X. Fu, B. Tang, M. Pecht, Deep residual shrinkage networks for fault diagnosis, IEEE Trans. Industr. Inf. 16 (2019) 4681–4690.
[8] W. Zhang, C. Li, G. Peng, Y. Chen, Z. Zhang, A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load, Mech. Syst. Signal Process. 100 (2018) 439–453.
[9] S. Khan, T. Yairi, A review on the application of deep learning in system health management, Mech. Syst. Signal Process. 107 (2018) 241–265.
[10] B. Yang, Y. Lei, F. Jia, N. Li, Z. Du, A polynomial kernel induced distance metric to improve deep transfer learning for fault diagnosis of machines, IEEE Trans. Industr. Electron. 67 (2019) 9747–9757.
[11] R. Xie, K. Hao, B. Huang, L. Chen, X. Cai, Data-driven modeling based on two-stream λ) gated recurrent unit network with soft sensor application, IEEE Trans. Industr. Electron. 67 (2019) 7034–7043.
[12] T. Han, C. Liu, L. Wu, S. Sarkar, D. Jiang, An adaptive spatiotemporal feature learning approach for fault diagnosis in complex systems, Mech. Syst. Signal Process. 117 (2019) 170–187.
[13] J. Lei, C. Liu, D. Jiang, Fault diagnosis of wind turbine based on long short-term memory networks, Renewable Energy 133 (2019) 422–432.
[14] A. Hard, C. Kiddon, D. Ramage, F. Beaufays, H. Eichner, K. Rao, R. Mathews, S. Augenstein, Federated learning for mobile keyboard prediction., arXiv preprint arXiv:1811.03604, 2018.
[15] T.S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I.C. Paschalidis, W. Shi, Federated learning of predictive models from federated electronic health records, Int. J. Med. Inform. 112 (2018) 59–67.
[16] S.R. Pokhrel, S. Singh, Compound-tcp performance for industry 4.0 wifi: A cognitive federated learning approach, IEEE Trans. Industr. Inf. 17 (2020) 2143–2151.

[17] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, et al., Communication-efficient learning of deep networks from decentralized data, arXiv preprint arXiv:1602.05629, 2016.
[18] J. Konečnỳ, H.B. McMahan, F.X. Yu, P. Richtárik, A.T. Suresh, D. Bacon, Federated learning: Strategies for improving communication efficiency, arXiv preprint arXiv:1610.05492, 2016.
[19] C. Zhang, P. Patras, H. Haddadi, Deep learning in mobile and wireless networking: a survey, IEEE Commun. Surv. Tutor. 21 (2019) 2224–2287.
[20] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, J. Zhang, Edge intelligence: Paving the last mile of artificial intelligence with edge computing, arXiv preprint arXiv:1905.10083, 2019.
[21] M. Blot, D. Picard, M. Cord, N. Thome, Gossip training for deep learning, arXiv preprint arXiv:1611.09726, 2016.
[22] X. Wang, Y. Li, T. Rui, H. Zhu, J. Fei, Bearing fault diagnosis method based on hilbert envelope spectrum and deep belief network, J. Vibroeng. 17 (2015) 1295–1308.
[23] Z. Chen, C. Li, R.V. Sánchez, Multi-layer neural network with deep belief network for gearbox fault diagnosis, J. Vibroeng. 17 (2015) 2379–2392.
[24] O. Janssens, V. Slavkovikj, B. Vervisch, K. Stockman, M. Loccufier, S. Verstockt, R.V.D. Walle, S.V. Hoecke, Convolutional neural network based fault detection for rotating machinery, J. Sound Vib. 377 (2016) 331–345.
[25] N. Talebi, M.A. Sadrnia, A. Darabi, Robust fault detection of wind energy conversion systems based on dynamic neural networks, Comput. Intell. Neurosci. 2014 (2014) 580972.
[26] H. Wang, Z. Liu, D. Peng, Y. Qin, Understanding and learning discriminant features based on multi-attention 1dcnn for wheelset bearing fault diagnosis, IEEE Trans. Industr. Inf. 16 (2019) 5735–5745.
[27] H. Lee, H. Jeong, G. Koo, J. Ban, S.W. Kim, Attention rnn based severity estimation method for interturn short-circuit fault in pmsms, IEEE Trans. Industr. Electron. 68 (2020) 3445–3453.
[28] X. Yuan, L. Li, Y. Wang, Nonlinear dynamic soft sensor modeling with supervised long short-term memory network, IEEE Trans. Industr. Inf. 16 (2019) 3168–3176.
[29] S. Hochreiter, J. Schmidhuber, Long short-term memory, Springer, Berlin Heidelberg, 1997.
[30] Z. Jiang, A. Balu, C. Hegde, S. Sarkar, Collaborative deep learning in fixed topology networks, Advances in Neural Information Processing Systems (2017) 5904–5914.
[31] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980, 2014.
[32] C. Liu, D. Jiang, W. Yang, Global geometric similarity scheme for feature selection in fault diagnosis, Expert Syst. Appl. 41 (2014) 3585–3595.
[33] P.F. Odgaard, J. Stoustrup, M. Kinnaert, Fault-tolerant control of wind turbines: a benchmark model, IEEE Trans. Control Syst. Technol. 21 (2013) 1168–1182.
[34] C. Lessmeier, J.K. Kimotho, D. Zimmer, W. Sextro, Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: a benchmark data set for data-driven classification, in: Proceedings of the European conference of the prognostics and health management society, 2016, pp. 05–08.
[35] M. Ruiz, L.E. Mujica, S. Alferez, L. Acho, C. Tutiven, Y. Vidal, J. Rodellar, F. Pozo, Wind turbine fault detection and classification by means of image texture analysis, Mech. Syst. Signal Process. 107 (2018) 149–167.
[36] P.F. Odgaard, J. Stoustrup, M. Kinnaert, Fault tolerant control of wind turbines – a benchmark model, IFAC Proceedings 42 (2009) 155–160.