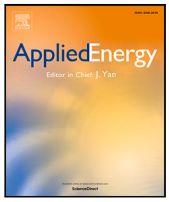
Applied Energy 377 (2025) 124532

Contents lists available at ScienceDirect

Applied Energy

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



A clustered federated learning framework for collaborative fault diagnosis of wind turbines

Rui Zhou [a], Yanting Li [a,*], Xinhua Lin [b]

[a] Department of Industrial Engineering and Management, Shanghai Jiao Tong University, Shanghai, China
[b] High-Performance Computing Center, Shanghai Jiao Tong University, Shanghai, China

ARTICLE INFO

Keywords:
Wind turbine
Fault diagnosis
Federated learning
Data heterogeneity
Model similarity

ABSTRACT

Data-driven approaches demonstrate significant potential in accurately diagnosing faults in wind turbines. To enhance diagnostic performance and reduce communication costs in federated learning with data heterogeneity among different clients, we introduce a clustered federated learning framework to wind turbine fault diagnosis. Initially, a lightweight multiscale separable residual network (LMSRN) model is proposed for each local client. The LMSRN model integrates a multiscale spatial feature derivation unit and a depthwise separable feature extraction unit. Subsequently, to tackle data heterogeneity among clients, canonical correlation coefficients of representations are extracted from the intermediate layers of local LMSRN models, and a representational canonical correlation clustering (RCCC) method is proposed to assess the similarity of local LMSRN models and group them into clusters. Finally, a global model is trained for each cluster. Real-world wind turbine data experiments showcase the superior performance of the proposed clustered federated learning framework over traditional methods in terms of diagnostic accuracy and computational speed. Additionally, the optimal choice of the number of clusters is also discussed.

1. Introduction

Wind energy, known for being pollution-free, eco-friendly, and self-renewable, is considered a clean and sustainable natural resource. When combined with solar and hydropower, it has the potential to economically and technologically improve global energy supply. The wind energy market has shown consistent year-on-year growth in total installed capacity. As of the end of 2022, the global total installed capacity has reached 906 GW, with projections to rise to 1221 GW by the end of 2030, making wind energy the fastest-growing sector in renewable energy generation capacity [1]. The International Energy Agency's (IEA) 2023 report indicates a steady increase in global wind energy production, with a record-breaking 265 TWh increase in 2022 [2].

Despite being one of the most promising renewable energy sources, wind power faces numerous challenges. The high cost of operation and maintenance already accounts for 30% or more of the total life-cycle cost [3]. The uncertainty of turbine health in complex environments and maintenance challenges are pushing operators towards more predictive and proactive decision-making. Large turbines are always equipped with SCADA systems that record immense operation data, environmental conditions, fault alarms, and event logs every 10 min. SCADA data can improve operators' remote performance monitoring and fault diagnosis [4,5].

In the field of fault diagnosis, deep learning, as a branch of machine learning, is widely applied in both academia and industry due to its remarkable feature learning capabilities and capacity for handling large-sample high-dimensional data [6,7]. Jia et al. (2016) [8] developed a deep neural network (DNN) capable of extracting nonlinear functions for diagnosing rolling bearings. Bach-Andersen et al. (2019) [9] proposed a deep learning model using convolutional neural networks for monitoring large-scale wind turbine drivetrain systems, validated on 251 wind turbine bearings. Liu et al. (2018) [10] utilized recurrent neural networks (RNN) to capture temporal correlation in signals. To ensure model performance for these techniques, sufficient labeled data is crucial. However, wind turbines often have low failure rates annually, resulting in a scarcity of failure data and an imbalance in positive and negative labels. Furthermore, newer wind farms have less data compared to older ones due to shorter operational periods.

Transfer learning, a machine learning technique enabling the transfer of diagnostic knowledge across different domains, can address these issues by sharing data among turbines and wind farms to mitigate class imbalance and sample size challenges. Tong et al. (2019) [11] observed significant differences between training and testing data in

* Corresponding author.
E-mail address: ytli@sjtu.edu.cn (Y. Li).

https://doi.org/10.1016/j.apenergy.2024.124532
Received 19 March 2024; Received in revised form 10 September 2024; Accepted 16 September 2024
Available online 25 September 2024

various working conditions, impacting model performance. As a result, they proposed a feature transfer learning method to address disparities in the conditional and marginal distributions of the source domain data. Yang et al. (2019) [12] conducted simulations on bearing faults to gain diagnostic knowledge of laboratory-used motor bearings. They utilized a convolutional neural network (CNN) to extract transferable features for diagnosing real-case bearings. Chen et al. (2021) [13] used transfer learning algorithms to calibrate data labels and evaluated model performance by analyzing gear cog belt fractures. Li et al. (2021) [14] integrated a convolutional autoencoder into a small-scale model and combined it with a parameter-based transfer learning framework to incorporate operational data from other wind turbines. Addressing the challenge of limited labeled data in new wind farms, Zhang et al. (2022) [15] introduced a balanced joint adaptive network (BJAN) to transfer data from other wind farms to the target one. They also developed a pseudo-label prediction method to balance labeled and unlabeled data. Overall, intelligent learning methods that leverage data sharing can achieve satisfactory diagnostic performance, especially in scenarios with insufficient labeled data in the target wind farm. Additionally, these methods can effectively tackle issues like class imbalance and distribution disparities.

However, transfer learning based on data sharing may not ensure the confidentiality of wind farms' proprietary information. Owners of various wind farms are hesitant to share operation data of their turbines with other wind farm owners or store it on a centralized cloud server due to privacy concerns. This reluctance is particularly evident when it comes to sharing labeled fault data, as wind farm managers must safeguard against the disclosure of sensitive details during operations to prevent competitors from accessing valuable technical information and engaging in unfair practices.

To address data privacy concerns and overcome data isolation, Federated Learning (FL) has been introduced as a collaborative distributed machine learning approach in fault diagnosis [16]. The federated learning framework is shown in Fig. 1. Consider a scenario where a central server collaborates with N clients, each representing a wind farm. Each client has its own local dataset, which is not shared with other clients or the server. The server has access only to the local model parameters from clients and utilizes these parameters through multiple communication rounds to develop an effective global diagnostic model. Assuming the server schedules $\mathcal{R}$ communication rounds for the federated learning task, let θ denote the trainable parameters of the diagnostic model. At the beginning of tth communication round, the central server broadcasts the global model parameters $\theta^{(t)}$ to N clients to initialize the local models for that round's iteration. Subsequently, clients independently train their local models using preprocessed local data. When local epoch reaches the upper limit or the local model converges, each client uploads local model parameters to the server. Upon receiving parameters $\left\{\theta_1^{(t)}, \theta_2^{(t)}, \ldots, \theta_N^{(t)}\right\}$ the server applies a specific model aggregation algorithm $\mathcal{A}(\cdot)$ to process the parameters and obtain the global parameters $\theta^{(t+1)}$ for next communication round. The wind turbine fault diagnosis models based on federated learning have achieved good performance while protecting the privacy and security of data from various regional wind farms. Jiang et al. [17] designed a residual convolutional network with self-attention mechanism, and completed the client-side model aggregation under the framework of deep federated learning (DeepFedWT). Cheng et al. [18] proposed to solve the imbalance of wind turbine icing data according to client characteristics, and built a global weighted federated learning model.

The most commonly used model aggregation algorithm is FedAvg. Global model parameters are the weighted average of the clients' parameters [19]. FedAvg does not consider the heterogeneity across clients, resulting in a global model with parameter distribution and performance that may not align with the data distribution characteristics of different clients. Karimireddy et al. [20] proved that applying FedAvg for parameter updates causes "drift" in client models, where some client models deviate from the global optimum.

To tackle this challenge, researchers have explored various model aggregation algorithms to reduce the effects of data heterogeneity and improve overall robustness. In [20], stochastic controlled averaging algorithm (SCAFFOLD) is proposed to correct this drift by estimating the differences in update directions between the global model and local models. Li et al. [21] added a proximal term in the client's objective function, and the proposed FedProx applied regularization to improve stable convergence in non-IID scenarios. Some scholars have considered the similarity of data and models between different clients and introduced clustering in the federated learning framework when dealing with non-IID data. Briggs et al. [22] introduced hierarchical clustering after training a global model, and partitioned clients into different groups according to the similarities between local models and the global model. Clients trained their group models locally in parallel. Palihawadana et al. [23] reduced the dimensions of collected local model parameters for distance clustering to adapt the aggregation weights of updating global parameters.

However, they still have the following drawbacks: First, the deep fault diagnosis models used by clients have excessive capacity and numerous parameters, reducing training efficiency. This is particularly challenging for edge devices like wind turbines, as overly complex models require operators to invest more in updating equipment performance. Furthermore, in each communication round, all clients are required to download the model from the central server and upload model parameters or gradients, incurring significant communication costs due to the large parameter size. Secondly, the statistical heterogeneity in wind turbine operation data across different regions and seasons leads to varying parameter distribution and diagnostic mechanisms among clients' local models. Although many aggregation strategies have been proposed, accommodating this heterogeneity with a single global model remains a difficult task.

To tackle the challenges in wind turbine fault diagnosis using federated learning, this paper proposes a lightweight clustered federated learning framework. The framework leverages the lightweight multiscale separable residual network (LMSRN) to extract spatial features for fault diagnosis. In this framework, the server initiates LMSRN model training tasks for clients, who perform local training and upload model parameters while ensuring data privacy. The server then clusters clients using a model clustering algorithm based on representational canonical correlation analysis. Within each cluster, clients complete local training tasks and upload parameters to support model aggregation. The output of the server includes clusters based on client model characteristics and their corresponding cluster models. This approach addresses the heterogeneity in wind turbine operation data, leading to improved client similarity inference and fault diagnosis performance through efficient communication and targeted clustering algorithms. We make contributions in the following ways:

- Lightweight multiscale separable residual network (LMSRN): We propose a novel LMSRN model to extract multiscale features from SCADA data, which can effectively capture the complex fault patterns of wind turbines. In federated learning, frequent exchanges of model parameters between clients and servers, including updates, uploads, and downloads, lead to substantial communication costs. Given the limitations of edge devices, lightweight and efficient diagnostic models are more promising. This paper employs depthwise separable blocks in place of traditional CNN blocks, significantly reducing the model's parameter size and enhancing the efficiency of parameter transmission in resource-constrained environments.
- Clustered federated learning (CFL) framework for collaborative fault diagnosis: Wind farms often compete, making cross-farm data sharing challenging, while single wind farms may face data shortages and silos. Federated learning addresses these issues by integrating the diagnostic models of all wind farms through the sharing of local model parameters, resulting in a global optimal

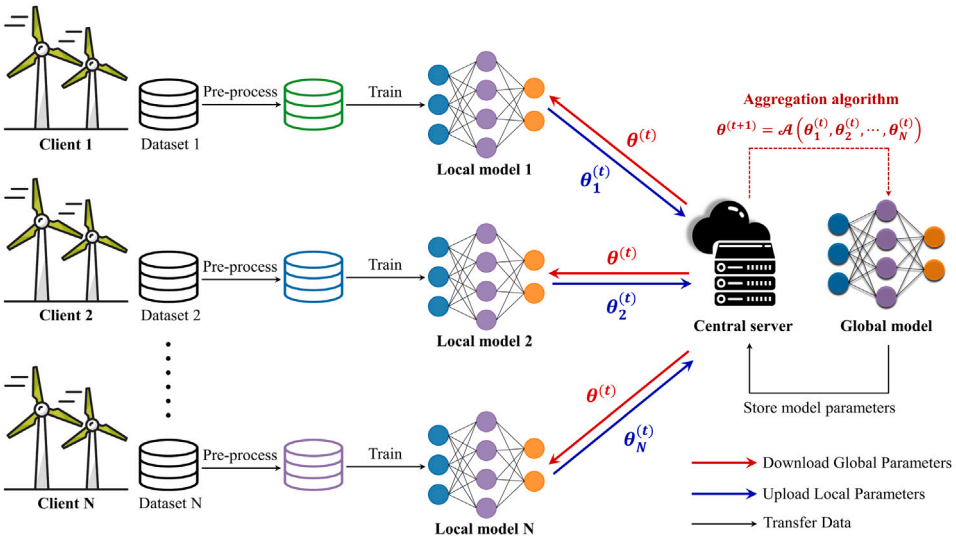


Fig. 1. The framework of federated learning with a server and multiple clients.

model while maintaining data privacy for each wind farm. The operational data of wind turbines from different wind farms is highly heterogeneous due to varying operational conditions, sensor configurations, and environmental factors. The data heterogeneity poses a significant challenge for traditional federated learning framework, which supports only a single global model on the server to perform diagnostic tasks for all wind farm clients. This global model, typically derived from the weighted averaging of local model parameters, often loses the unique data characteristics of individual wind farms, resulting in weaker detection capabilities for rare fault types. We propose a novel clustered federated learning framework designed to handle data heterogeneity effectively. By clustering clients based on model characteristics, it is ensured that similar clients are grouped together, leading to more homogeneous data within each cluster and improving the overall diagnostic performance.

- Representational canonical correlation clustering (RCCC): The core of the clustered federated learning framework proposed in this paper is using local model parameters to assess client similarity and thereby form clusters. Due to the large scale and strong sparsity of deep learning model parameters, traditional clustering methods fail to effectively represent model characteristics, leading to erroneous client clustering. Canonical correlation analysis (CCA) is a powerful method for measuring similarity between neural network models. Additionally, the intermediate layer representations of models serve as crucial indicators of model functionality. This paper introduces representational canonical correlation clustering (RCCC), which computes the canonical correlation coefficients of local model representations to determine client similarities, followed by spectral clustering to form the clusters.

The subsequent structure of this paper includes the introduction of real-world wind turbine datasets of various wind farms in Section 2, followed by an explanation of the data preprocessing process. Section 3 presents a clustered federated learning framework to wind turbine fault diagnosis, utilizing the lightweight multiscale separable residual network (LMSRN). The performance of the proposed method is evaluated on datasets from real wind farms in Section 4, and the test results are thoroughly analyzed. Section 5 concludes with a summary of the advantages and scope of application of the proposed method, along with future research directions.

2. Data description and preprocessing

2.1. Data description

The SCADA dataset analyzed in this study comprises operational data from wind turbines across 13 wind farms in various regions of China. Information for each wind farm is presented in Table 1. The data was collected from January 2019 to December 2019, capturing 68 variables at 10-minute intervals. These SCADA samples include: wind variables like wind speed, average wind speed in 30 s, wind direction angle, average wind direction angle in 60 s; temperature variables like ambient temperature and inside temperature of different components; electrical variables such as active and reactive power; and operating variables like cabin position, wind turbine speed, and pitch angle.

Fault logs provide information on the occurrence of different types of faults during specific time periods. The wind turbine faults are recorded using fault codes in logs, which capture the time from activation to reset. Most faults typically last around 10 min. Due to turbines mostly operating in a normal state, the amount of specific fault data available is limited. We categorize all specific faults into eight classes: gearbox fault, engine room fault, converter fault, pitch system fault, vibration sensor fault, generator fault, hydraulic system fault, and yaw system fault. Table 2 displays the fault categories and some specific faults. There is a notable imbalance in data quantity across different faults. In Fig. 2, the gray star-filled bars represent the amount of downsampled normal data points while the colored bars represent the original amount of fault data points collected from 13 wind farms. The fault data demonstrate strong heterogeneity since turbines of different wind farms show varying fault types and corresponding quantities of fault data.

- The prevalence of gearbox fault, converter fault, pitch system fault, vibration sensor fault, and generator fault varies across wind farms. Notably, WF8, WF10, and WF13 are free of gearbox fault, while WF2, WF6, WF7, WF8, and WF13 do not have converter fault. Generator fault is absent in WF1, WF4, WF6, WF7, WF8, and WF10. Pitch system fault and vibration sensor fault are less common, with WF3, and WF4 showing higher instances of pitch system fault, and WF1, and WF5 exhibiting more vibration sensor faults.
- Certain wind farms are characterized by specific fault types. For instance, WF1 has a small amount of engine room faults, converter faults, vibration sensor faults, and hydraulic system faults, while the number of gearbox faults and yaw system faults is

Table 1
Details of the selected wind farms.

Wind farm	Location	No. Turbines	Rated power	Collection time
WF1	Jiangsu, China	25	2.0 MW	from 01/01/2019 to 25/12/2019
WF2	Yunnan, China	24	2.0 MW	from 01/01/2019 to 25/12/2019
WF3	Tianjin, China	19	2.0 MW	from 01/01/2019 to 25/12/2019
WF4	Tianjin, China	19	2.0 MW	from 01/01/2019 to 25/12/2019
WF5	Shanghai, China	25	3.6 MW	from 01/01/2019 to 25/12/2019
WF6	Shanghai, China	25	3.6 MW	from 01/01/2019 to 25/12/2019
WF7	Jiangsu, China	25	3.6 MW	from 01/01/2019 to 25/12/2019
WF8	Jiangsu, China	28	3.6 MW	from 01/01/2019 to 27/12/2019
WF9	Shanghai, China	26	3.6 MW	from 01/01/2019 to 27/12/2019
WF10	Shanghai, China	26	3.6 MW	from 01/01/2019 to 25/12/2019
WF11	Zhejiang, China	25	3.6 MW	from 01/01/2019 to 25/12/2019
WF12	Hubei, China	26	2.0 MW	from 01/01/2019 to 25/12/2019
WF13	Hubei, China	19	2.0 MW	from 01/01/2019 to 25/12/2019

Table 2
Fault categories and some specific faults.

Class	Fault categories	Examples of specific faults
1	Gearbox fault	Gearbox cooling water pressure failure Gearbox lubricating oil pump motor protection
2	Engine room fault	Engine room cooling fan protection Engine room specific module failure
3	Converter fault	Converter fails and shuts down Converter ups alarms
4	Pitch system fault	Pitch system communication failure Pitch system shaft drive failure
5	Vibration sensor fault	Vibration sensor communication failure Vibration sensor missing
6	Generator fault	Generator cooling water temperature exceeds limit Generator cooling water pressure failure
7	Hydraulic system fault	Hydraulic oil pump motor protection Hydraulic oil pump heater protection
8	Yaw system fault	Yaw system motor protection Yaw system soft starter slope error

higher. WF3 shows a higher quantity of gearbox faults, converter faults, and pitch system faults. WF4 has a significant number of faults in the engine room, converter, and pitch system. WF6 and WF7 have notable gearbox faults, engine room faults, and yaw system faults, while WF8 has merely experienced engine room faults and yaw system faults. WF10 has faults related to the engine room, converter, and yaw system. WF12 has limited pitch system faults and hydraulic system faults.

- There is a significant disparity in the amount of data across different fault categories. For example, WF12 has recorded over 1000 engine room faults, but the number of pitch system and hydraulic system faults is less than 100. WF2 has only 37 gearbox faults compared to 865 engine room faults. WF5 has data for all fault types except pitch system faults, with the highest volume of yaw system fault data being 1518, while the lowest volume of vibration sensor fault data is just 34.

There are notable variations in the distribution of SCADA variables across different wind turbines during faults. Internal temperature of the generator, generator speed, and active power are three critical variables for fault diagnosis. After aggregating all the SCADA data of wind turbines from the same wind farm, Fig. 3 demonstrates considerable heterogeneity in the distribution of these variables across different wind farms.

The variability in the distribution of variables may indicate differences in the internal mechanisms of different wind turbines when encountering the same fault. Fig. 4 illustrates the power curves of turbines in various wind farms under three faults. In the event of gearbox faults, most turbines in WF12 consistently exhibit low power output. Conversely, most turbines in WF6 continue to function at moderate to high-power levels, while some turbines in WF4 operate at high power. For converter faults, turbines in WF5 are in low power operating mode, while a limited amount of turbines in WF3 and WF4 operate at medium power levels. Regarding hydraulic system faults, turbines in WF9 tend to operate across different power levels, whereas turbines in WF1 are in a low power output state, and turbines in WF2 are predominantly operating at medium power output levels.

Data heterogeneity is commonly found in federated fault diagnosis tasks involving multiple turbines and wind farms. Failing to account for this heterogeneity when aggregating client models can lead to decreased diagnostic performance and slower convergence of the global model.

2.2. Data preprocessing

The preprocessing workflow, illustrated in Fig. 5, includes variable screening, data partitioning, data balancing, and data normalization.

In order to effectively perform fault diagnosis, certain variables with constant values, such as capacitor voltage, generator slip ring temperature, and cooling fan outlet temperature, are removed as they may not significantly contribute to the iteration and training of local models. Variables like monthly power generation, which are not relevant to real-time operation processes, are also excluded. Table 3 summarizes 39 selected variables as input of local models.

Stratified data partition is essential to ensure consistent data distribution and to maintain the separation between model training and evaluation processes. The process initially entails downsampling the normal data within time windows, followed by partitioning the resultant time-series data encapsulated in these windows into a 70:30 ratio for training and testing, ensuring the preservation of temporal dynamics throughout the division. Subsequent data balancing on the training dataset increases the quantity ratio to around 75:25. It is important to note that the data is partitioned first to keep the test dataset authentic and free from any influence during the model training phase.

After stratified sampling to split the training and test data, the proportion of fault to normal data in the training dataset is skewed. Directly inputting such dataset into the deep network for model training may result in severe overfitting, making data balancing crucial. To address this issue, we adjusted the proportion by utilizing the Synthetic Minority Oversampling Technique (SMOTE) [24] to oversample the fault data in the training dataset. In Table 4, a comparison is conducted on the data quantity of positive and negative timestamped samples in the datasets before and after balancing, encompassing all eight types of faults. It is clear that after balancing, the class distribution of the data is more balanced, allowing client-trained local models to accurately predict the minority class.

In order to remove differences in numerical values and the influence of units among different variables, we need to apply a normalization function to scale each variable in the input dataset. The normalized

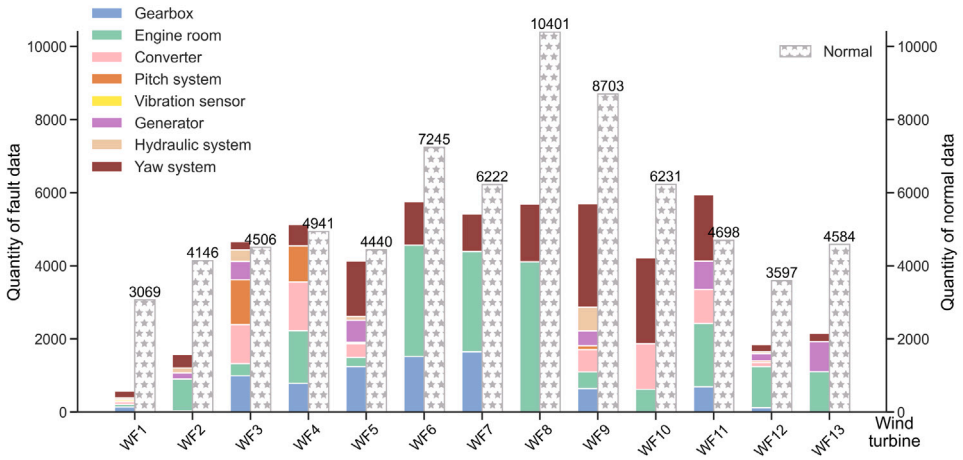


Fig. 2. Class distribution of each wind turbine. Left *y*-axis: the amount of fault data points. Right *y*-axis: the amount of downsampled normal data points.

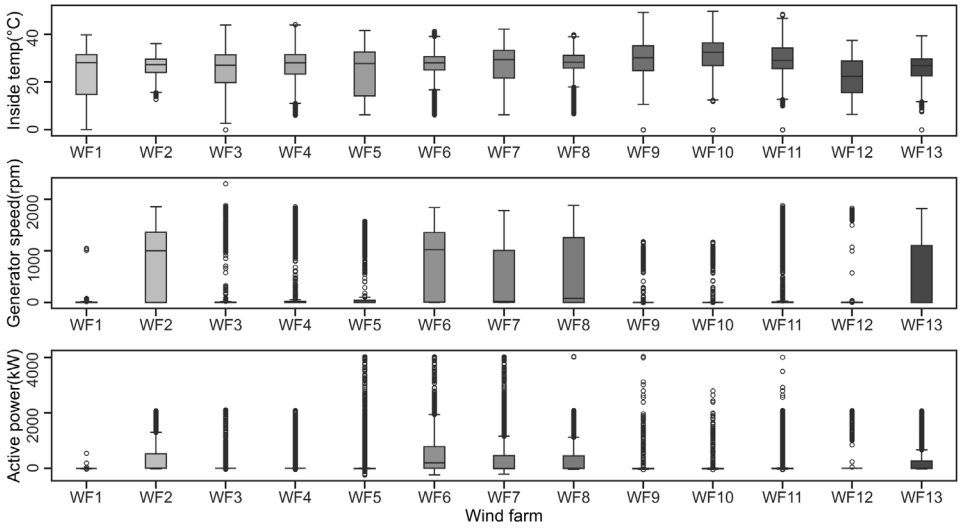


Fig. 3. Distribution of inside temp (°C), generator speed (rpm), and active power (kW) across all the wind farms at faults.

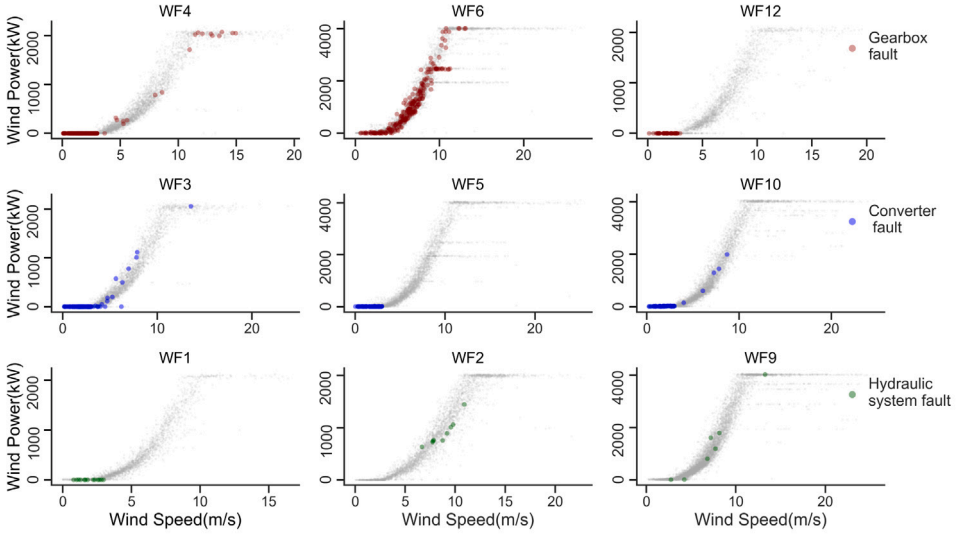


Fig. 4. The disparities in power curves among different wind farms encountering different faults. Gearbox faults (WF4, WF6, and WF12), converter faults (WF3, WF5, and WF10), hydraulic system faults (WF1, WF2, and WF9).

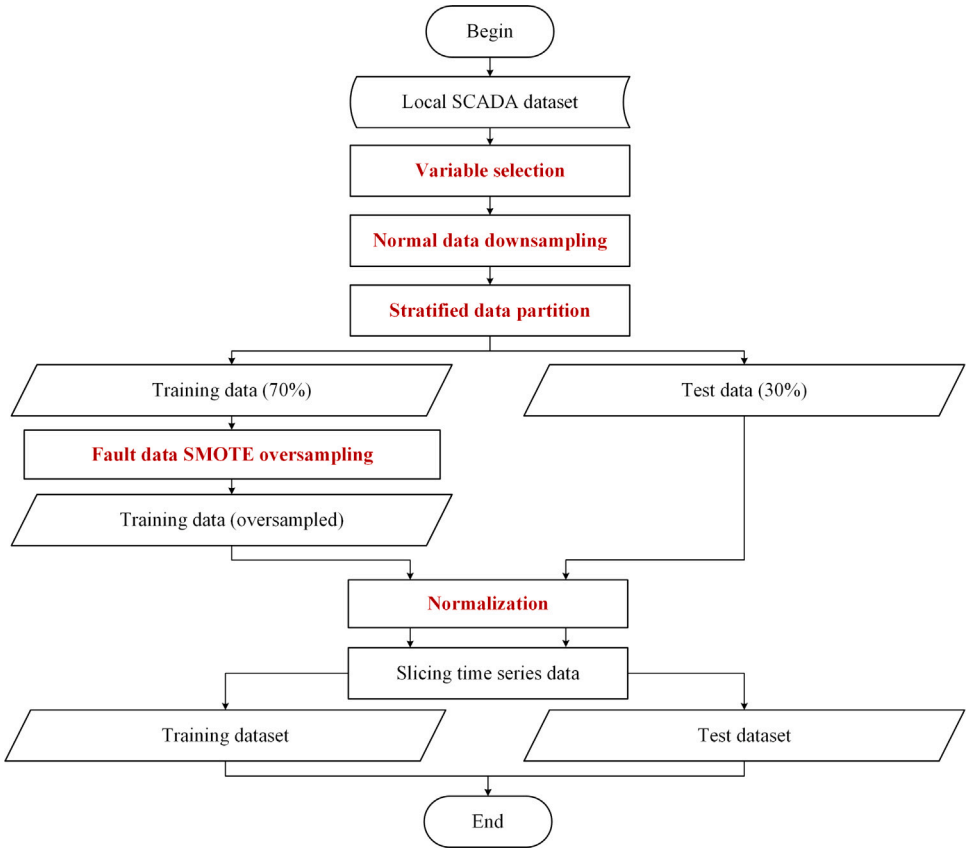


Fig. 5. Workflow of local SCADA data preprocessing.

Table 3
Variables of operation data used as model input.

ID	Variable name	ID	Variable name	ID	Variable name
1	instantaneous wind speed	14	power factor	27	outside temp
2	average wind speed in 30s	15	set value of active power	28	inside temp
3	wind direction angle	16	set value of power factor	29	tower drum temp
4	average wind direction angle in 60s	17	low-speed bearing temp	30	bottom cabinet temp
5	grid A phase voltage	18	high-speed bearing temp	31	engine room cabinet temp
6	grid B phase voltage	19	free end bearing temp	32	cabin position
7	grid C phase voltage	20	drive end bearing temp	33	twisting cable position
8	grid A phase current	21	gearbox oil temp	34	wind turbine speed
9	grid B phase current	22	gearbox cooling water temp	35	generator speed
10	grid C phase current	23	generator cooling water temp	36	hydraulic system pressure
11	daily availability	24	generator stator U temp	37	pitch angle
12	active power	25	generator stator V temp	38	vibration value of cabin X
13	reactive power	26	generator stator W temp	39	vibration value of cabin Y

SCADA variable is shown as $\frac{x-\mu}{\sigma}$ where μ and σ denote the mean and standard error of x.

To ensure consistency and temporal ordering in model inputs, we utilize sliding windows of identical duration to slice the sample segments. Given that the collecting interval of SCADA data is 10 min, it is reasonable to avoid setting windows with excessively large spans. In this paper, we therefore opt for a window size of 3, which strikes a balance between capturing meaningful temporal dynamics without overly diluting the temporal resolution inherent in the data. This approach maintains the sequential nature of the data, allowing the model to learn from patterns that evolve over fixed time intervals.

3. Methodology

Assume that the number of clients is N, and the ith client has its own raw dataset D_i, which is not released to the public. The local datasets of all clients define a set $\{D_1, D_2, \ldots, D_N\}$. The ith client simply uses local dataset D_i with a sample size of n_i to train the local model $\mathcal{M}_i$, and the objective is to minimize local loss function $\ell_i(\theta)$, where θ represents the model parameters that need to be optimized.

As shown in Fig. 6, we propose a clustered federated learning framework to fault diagnosis of wind turbines. The framework involves initial training of local lightweight multiscale separable residual network models by each client, followed by partitioning clients into clusters based on the similarity of the parameters of their local model. Denote the collection of clusters as $C = \{C_1, \ldots, C_{|C|}\}$. Thirdly, a hybrid federated model $\mathcal{M}^j, j = 1, \ldots, |C|$, is build for cluster j and assigned to every client in cluster j. Lastly, In each communication round, the ith client in cluster j downloads the cluster model $\mathcal{M}^j$ and locally train it to get the model $\mathcal{M}_i^j$. The optimization objective of clustered federated learning is to minimize the objective function:

$$\mathcal{L}_j(\theta) = \sum_{i \in C_j} \frac{n_i}{n_j} \ell_i(\theta), \forall j \in \{1, 2, \ldots, |C|\} \quad (1)$$

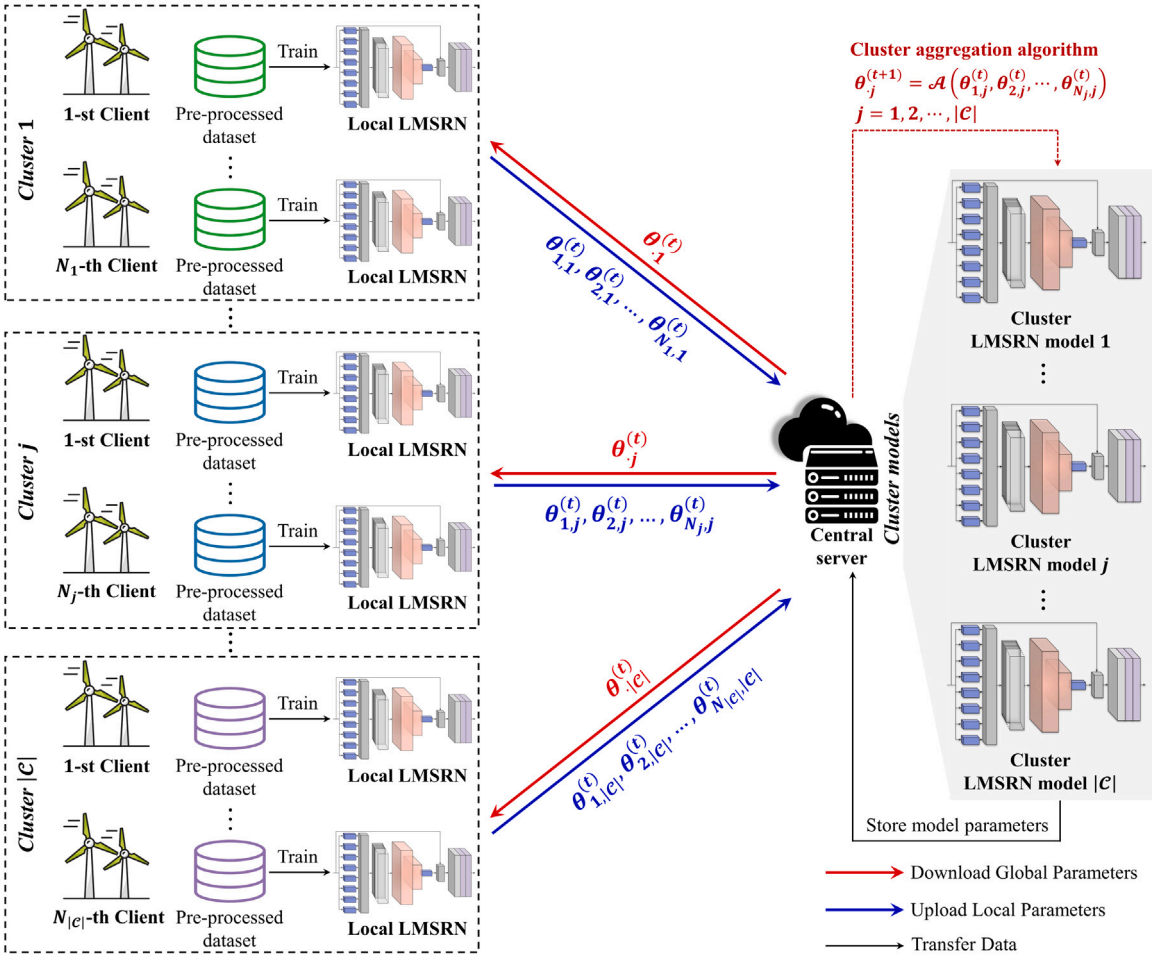


Fig. 6. The framework of clustered federated learning.

Table 4
Quantity of original and balanced datasets (data points in timestamp format) for 13 wind farms in different states.

Wind farm	Original dataset		Balanced dataset	
	Fault (positives)	Normal (negatives)	Fault (positives)	Normal (negatives)
WF1	575	$> 10^5$	3677	3069
WF2	1576	$> 10^5$	5089	4146
WF3	4663	$> 10^5$	9408	4506
WF4	5136	$> 10^5$	7146	4941
WF5	4136	$> 10^5$	7916	4440
WF6	5757	$> 10^5$	7372	7245
WF7	5418	$> 10^5$	6863	6222
WF8	5689	$> 10^5$	7840	10 401
WF9	5703	$> 10^5$	11 352	8703
WF10	4225	$> 10^5$	6005	6231
WF11	5944	$> 10^5$	8186	4698
WF12	1846	$> 10^5$	6127	3597
WF13	2153	$> 10^5$	4001	4584

where $n_j = \sum_{i \in C_j} n_i$ represents the entire sample sizes of cluster j. In essence, all clusters train their own cluster models in parallel, with the aim of optimizing their respective objective functions.

3.1. Lightweight multiscale separable residual network

A lightweight multiscale separable residual network (LMSRN) is introduced in this section, as illustrated in Fig. 7. The network utilizes a multiscale spatial feature derivation unit, incorporating multiple convolution kernels in the same layer to enhance spatial information dimensions and generate compact feature maps. Additionally, it employs a depthwise separable feature extraction unit to separate spatial and channel information of the feature maps, improving feature extraction and reducing network parameters. LMSRN demonstrates reduced parameters and network complexity, effectively preventing overfitting and maintaining diagnostic accuracy. When integrated into the federated learning framework as local and global models, it helps decrease communication costs per round to some extent.

3.1.1. Multiscale spatial feature derivation unit

Zhang et al. [25] proposed a multiscale feature extraction unit to reduce network depth while improving feature extraction capabilities, which inspired us to develop the multiscale spatial feature derivation unit.

The multiscale spatial feature derivation unit involves cloning input samples n^1 times and passing them through eight Conv2D blocks with the same number of output channels to generate n^1 sets of feature maps. By using Conv2D with different kernel sizes, the unit can extract spatial features of varying scales and enhance the distinguishability of feature maps. In contrast to the multiscale feature extraction unit, the multiscale spatial feature derivation unit performs concatenation in the spatial dimension to achieve spatial feature derivation.

SCADA data $\boldsymbol{x} \in \mathbb{R}^{V \times T}$ is fed into the network. Output of the multiscale spatial feature derivation unit can be expressed as follows:

$$\boldsymbol{h}^1 = SpatialConcat\left\{ReLU\left(\boldsymbol{W}_1^1 \boldsymbol{x} + \boldsymbol{B}_1^1\right), \ldots, ReLU\left(\boldsymbol{W}_{n^1}^1 \boldsymbol{x} + \boldsymbol{B}_{n^1}^1\right)\right\} \tag{2}$$

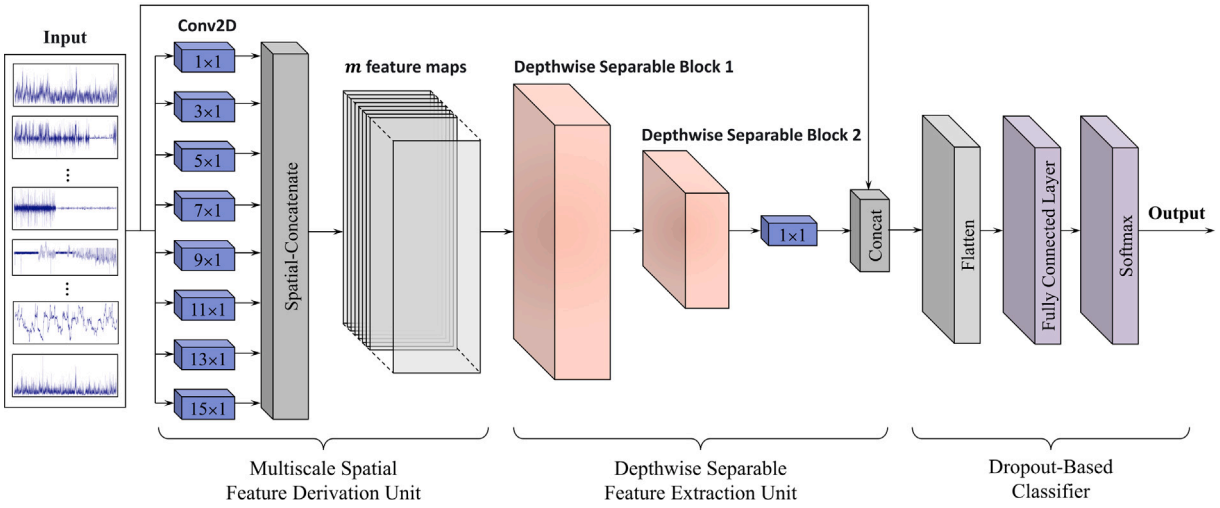


Fig. 7. Overall structure of the LMSRN.

where the ith Conv2D with a multiscale kernel size of $(2i-1,1)$ denotes the weight matrix and bias as $\boldsymbol{W}_i^1$ and $\boldsymbol{B}_i^1$, $i = 1,2,\ldots,n^1$. $ReLU(\cdot)$ is the rectified linear activation function [26]. $SpatialConcat\{\cdot\}$ subsequently combines feature maps output by each Conv2D in the spatial dimension.

3.1.2. Depthwise separable feature extraction unit

Inspired by MobileNetV2 [27,28], and Xception [29], which implemented a separable dense block to eliminate redundant information from the feature maps, we proposed two depthwise separable blocks with different convolutional kernel sizes for feature extraction to reduce the number of parameters.

The proposed depthwise separable block reinterprets conventional convolution as a combination of pointwise convolution and depthwise convolution. The depthwise convolution focuses on capturing spatial relevance, while the pointwise convolution is responsible for capturing cross-feature map (channel-level) relevance. In the first depthwise separable block illustrated in Fig. 8, a smaller depthwise convolution kernel (3 × 3) is used to extract features from a smaller spatial range and simultaneously reduce the output feature maps' channel dimensions. This process effectively reduces the number of parameters and computational complexity.

As the input of this block, $\boldsymbol{h}^1$ denotes the feature maps with m channels, and is operated by a pointwise convolution with 1 × 1 kernelsize to reconstruct m_1-channel feature maps, denoted by $\boldsymbol{h}^2 = ReLU\left(\boldsymbol{W}^2\boldsymbol{h}^1 + \boldsymbol{B}^2\right)$. Afterwards, $\boldsymbol{h}^2$ is split into m_1 independent channels $\left\{\boldsymbol{h}_\varphi^2\right\}_{\varphi=1}^{m_1}$, and the spatial relevance of each feature map is separately captured via depthwise convolution with m_1 filters. The φ-th channel of feature map is operated by φ-th depthwise filter to output:

$$\boldsymbol{h}_\varphi^3 = ReLU\left(\boldsymbol{W}_\varphi^3\boldsymbol{h}_\varphi^2 + \boldsymbol{B}_\varphi^3\right) \tag{3}$$

where $\boldsymbol{W}_\varphi^3$ and $\boldsymbol{B}_\varphi^3$ represent the weight matrix and bias of the φ-th filter respectively.

Subsequently, all the output feature maps $\left\{\boldsymbol{h}_\varphi^3\right\}_{\varphi=1}^{m_1}$ are combined as input $\boldsymbol{h}^3$ of the second depthwise separable block with a larger depthwise convolution kernel (5 × 5), thereby enlarging the receptive field to prevent the omission of long-range spatial features. In specific, $\boldsymbol{h}^3$ is operated by a pointwise convolution to reconstruct m_2-channel feature maps, denoted by $\boldsymbol{h}^4 = ReLU\left(\boldsymbol{W}^4\boldsymbol{h}^3 + \boldsymbol{B}^4\right)$. The second depthwise convolution with m_2 filters is then applied to operate $\left\{\boldsymbol{h}_\varphi^4\right\}_{\varphi=1}^{m_2}$ split from $\boldsymbol{h}^4$. With the weight matrix $\boldsymbol{W}_\varphi^5$ and bias $\boldsymbol{B}_\varphi^5$, the φ-th depthwise filter outputs $\boldsymbol{h}_\varphi^5 = ReLU\left(\boldsymbol{W}_\varphi^5\boldsymbol{h}_\varphi^4 + \boldsymbol{B}_\varphi^5\right)$. The combined feature maps $\left\{\boldsymbol{h}_\varphi^5\right\}_{\varphi=1}^{m_2}$ are then inputted into a 1 × 1 convolution to fuse the features and output $\boldsymbol{h}^6$.

Depthwise separable block has fewer parameters compared to conventional convolution with the same kernel size. The number of parameters of conventional convolution with kernel size K_{conv} are:

$$Params_{conv} = K_{conv} \times N_{in} \times N_{out} \tag{4}$$

where N_{in} is the channel dimension of input feature map and N_{out} is the channels of output. The number of depthwise separable blocks can be calculated as the sum of the parameters from pointwise convolution and depthwise convolution.

$$Params = N_{in} \times N_{out} + N_{out} \times K_{conv} = \left(K_{conv} + N_{in}\right) \times N_{out} \tag{5}$$

The pointwise convolution with 1 × 1 kernel controls the output channels. Its parameters are $Params_{point} = 1 \times N_{in} \times N_{out}$. The depthwise convolution with K_{conv} kernel maintains the channel dimension while extracting spatial features with N_{out} filters. Parameters of the depthwise convolution are calculated as $Params_{depth} = N_{out} \times K_{conv}$. As the number of feature maps increases, the parameters of depthwise separable blocks decrease geometrically.

3.1.3. Dropout-based classifier

The fused feature map $\boldsymbol{h}^6$ is concatenated with the residual $\boldsymbol{x}$ through the channel dimension, maintaining the original input's feature information at the channel level to prevent essential characteristics from being lost in the deep network. The flattened vector $\boldsymbol{\zeta}$ is then inputted into a dropout-based classifier for fault detection prediction. This classifier comprises fully-connected layers with dropout regularization, aimed at addressing overfitting problems in small-sample training situations and enhancing the model's generalization ability. Specifically, We define the predicted probability of ξ label as $\hat{y}^\xi$, where ξ ranges from 0 to K, with 0 representing the label for the normal state, while labels 1 to K denote the total types of faults. Additionally, $\hat{y}^\xi$ fits the following softmax function expression:

$$\hat{y}^\xi = \frac{\exp\left(\boldsymbol{w}^\xi\boldsymbol{\zeta}\right)}{\sum_{\xi=0}^{K}\exp\left(\boldsymbol{w}^\xi\boldsymbol{\zeta}\right)} \tag{6}$$

where $\boldsymbol{w}^\xi\boldsymbol{\zeta}$ denotes the ξ-th output factor of the fully connected layer. Consequently, the learning objective of each client is to minimize the local loss function $\ell(\boldsymbol{\theta})$, which is defined as the cross-entropy loss:

$$\ell(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{\xi=0}^{K} y_i^\xi \log\left(\hat{y}_i^\xi\right) \tag{7}$$

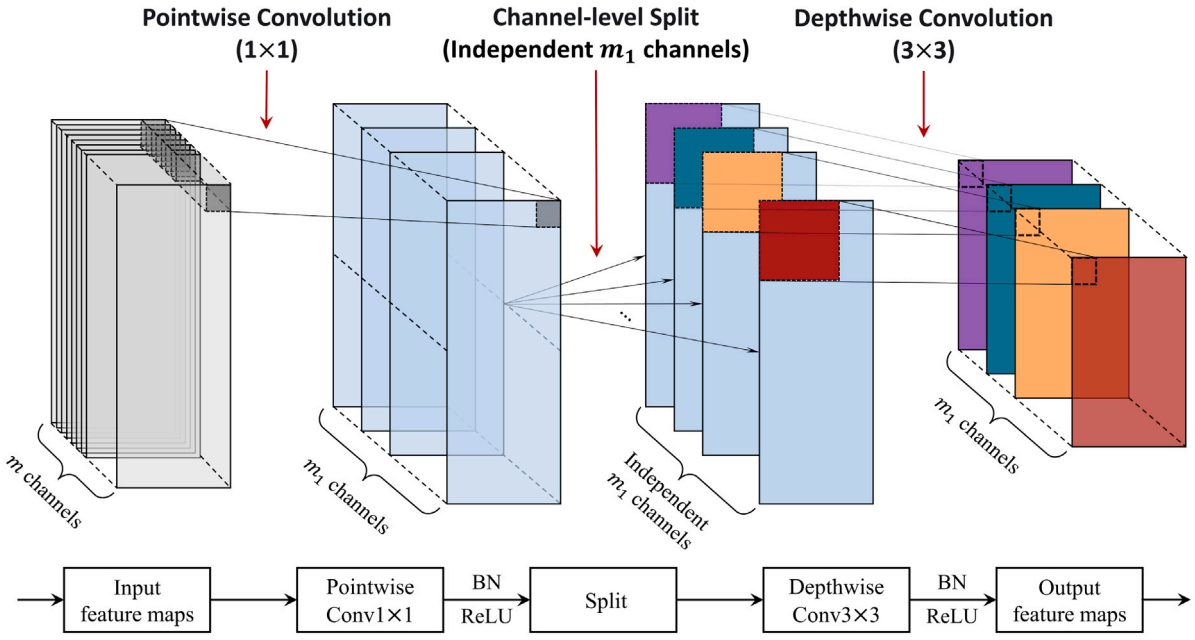


Fig. 8. Illustration of the depthwise block.

where n is the sample size of the local dataset, y_i^{ξ} is the true label of the ith sample, and $\hat{y}_i^{\xi}$ is the predicted probability of the ith sample for the ξ-th label.

3.2. Clustered federated learning

Following the development of local lightweight multiscale separable residual networks for each client, we proposed the representational canonical correlation clustering (RCCC) method to group these local models into distinct clusters. Initially, a representational similarity matrix is constructed through canonical correlation analysis (CCA) to measure the similarity between different local models. This matrix is then used to create a similarity graph that visually represents the relationships among the local models. By employing spectral clustering, the graph is partitioned to effectively divide the clients into separate clusters. Subsequently, the central server allocates cluster models to the clients within each cluster. During the subsequent communication rounds, clients retrieve the relevant cluster model parameters and carry out local training tasks. At the conclusion of each communication round, the server performs cluster-internal model aggregation.

The algorithm for clustered federated learning, as depicted in Algorithm 1, comprises three main modules: (1) local LMSRN model training, (2) representational canonical correlation clustering, and (3) cluster model training and model aggregation. The overall flowchart of the proposed clustered federated learning framework is illustrated in Fig. 9.

3.2.1. Local LMSRN model training

Prior to clustering clients into specific groups, each client conducts local LMSRN model training for a specified number of epochs, $\mathcal{E}$. And the initial local model parameters $\{\theta_i\}_{i=1}^{N}$ are obtained. Subsequently, the server gathers the parameters from all local LMSRN models.

3.2.2. Representational canonical correlation clustering

It is essential to obtain the representational similarity matrix derived from the local LMSRN models. The convolutional layers and fully connected layers within the model play significant roles in feature extraction and classification. Therefore, it is reasonable to compute the representations by utilizing the weights of both convolutional and fully connected layers.

An informative representation of a well-trained deep neural network ϕ measures the similarity among activations of a pre-defined input $\chi \in \mathbb{R}^{\eta \times V \times T}$ at certain layers ϕ^l. The pre-defined input can be obtained from publicly accessible datasets and do not necessitate a substantial sample size η. We define the representations of model ϕ at lth layer on given χ as follows:

$$\boldsymbol{R}^l \doteq \begin{pmatrix} \left(\phi^l \circ \phi^{l-1} \circ \cdots \circ \phi^1\right)(\chi_1) \\ \left(\phi^l \circ \phi^{l-1} \circ \cdots \circ \phi^1\right)(\chi_2) \\ \vdots \\ \left(\phi^l \circ \phi^{l-1} \circ \cdots \circ \phi^1\right)(\chi_\eta) \end{pmatrix} \in \mathbb{R}^{\eta \times d} \tag{8}$$

where $d \doteq d^l$ represents the quantity of neurons in layer l, and each sample point of χ is activated by layers $\left(\phi^l \circ \phi^{l-1} \circ \cdots \circ \phi^1\right)(\cdot)$.

For each client i, the weights of convolutional layers and fully connected layers are selected from θ_i and map the given χ into representations $\left\{\boldsymbol{R}_i^1, \boldsymbol{R}_i^2, \ldots, \boldsymbol{R}_i^L\right\}$, where L is the total number of convolutional and fully connected layers. For the sake of simplicity, we note the representations of local model $\mathcal{M}_i$ at layer l as $\boldsymbol{R}_i \doteq \boldsymbol{R}_i^l \in \mathbb{R}^{\eta \times d}$, where $l = 1, 2, \ldots, L$ and $d \doteq d^l$. Our goal is to quantify the similarity measurement $m\left(\mathcal{M}_i, \mathcal{M}_j\right)$ between the local models of any pair of clients, denoted as client i and client j, by extracting the correlations across their representations.

Existing research about the similarity of neural networks mainly emphasizes on the representational aspect of assessing how activations of intermediate neural layers differ [30]. Canonical correlation analysis (CCA) is a multivariate statistical method used to identify relationships between two sets of random variables [31]. Various approaches have been developed based on CCA to compare neural network representations and assess the similarity among different deep learning models [32–34]. This study presents a new method for determining the similarity of local LMSRN models, requiring no additional inputs.

Define the canonical weights $\omega_i \in \mathbb{R}^d$ and $\omega_j \in \mathbb{R}^d$ for client i and client j. Use the representations $\boldsymbol{R}_i$ and $\boldsymbol{R}_j$ as linear transformations to

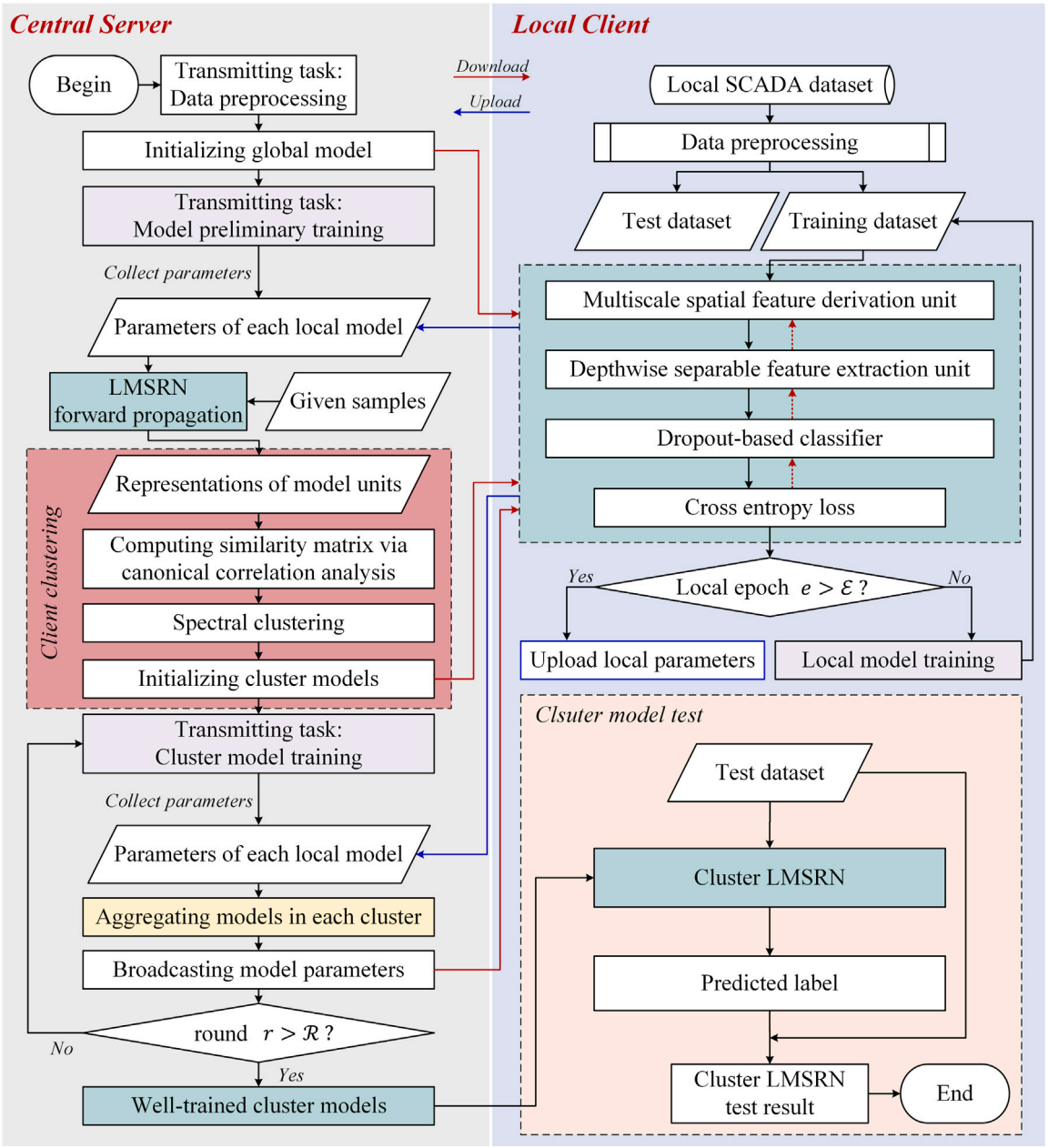


Fig. 9. Flowchart of the clustered federated learning.

project ω_i and ω_j in the space $\mathbb{R}^n$ such that the correlation $\rho_l\left(\boldsymbol{R}_i, \boldsymbol{R}_j\right)$ between the projections is maximized:

$$\rho_l\left(\boldsymbol{R}_i, \boldsymbol{R}_j\right) = \max_{\omega_i, \omega_j} \frac{\left\langle \boldsymbol{R}_i\omega_i, \boldsymbol{R}_j\omega_j \right\rangle}{\left\|\boldsymbol{R}_i\omega_i\right\| \cdot \left\|\boldsymbol{R}_i\omega_i\right\|} \tag{9}$$

A mean-based aggregation of the canonical correlations across all layers is used to compute the similarity measurement of local models:

$$m_{ij} \doteq m\left(\mathcal{M}_i, \mathcal{M}_j\right) = \frac{1}{L}\sum_{l=1}^{L} \rho_l\left(\boldsymbol{R}_i, \boldsymbol{R}_j\right) \tag{10}$$

where $i, j = 1, 2, \ldots, N$ and $i \neq j$. Obviously, m_{ij} is ranged from 0 to 1 and indicates greater similarity when near 1. Fig. 10 provides a detailed depiction of the similarity calculation process using examples of ith client and jth client. The procedure involves computing pairwise similarity measures between each pair of clients while iterating over all clients. As a result, a representational similarity matrix Δ is built by capturing m_{ij} as its element, which is essential to partition clients into various clusters via spectral clustering algorithm. To be more specific, the similarity matrix Δ is computed as follows:

$$\Delta = \begin{pmatrix} m_{11} & m_{12} & \cdots & m_{1N} \\ m_{21} & m_{22} & \cdots & m_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ m_{N1} & m_{N2} & \cdots & m_{NN} \end{pmatrix} \tag{11}$$

while $m_{ij} = m_{ji}$ and $m_{ii} = 1$. The detailed algorithm of similarity matrix computation based on RCCC is presented in Algorithm 2.

After obtaining the similarity matrix, proceed with the clustering process as follows. Using the similarity matrix $\Delta = \left(\Delta_{i,j}\right)_{N\times N}$ as a weighted adjacency matrix, construct an undirected weighted graph with the local models $\left\{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_N\right\}$ as the vertex set. Applied the spectral clustering algorithm to group local models with similar representational characteristics into the same cluster.

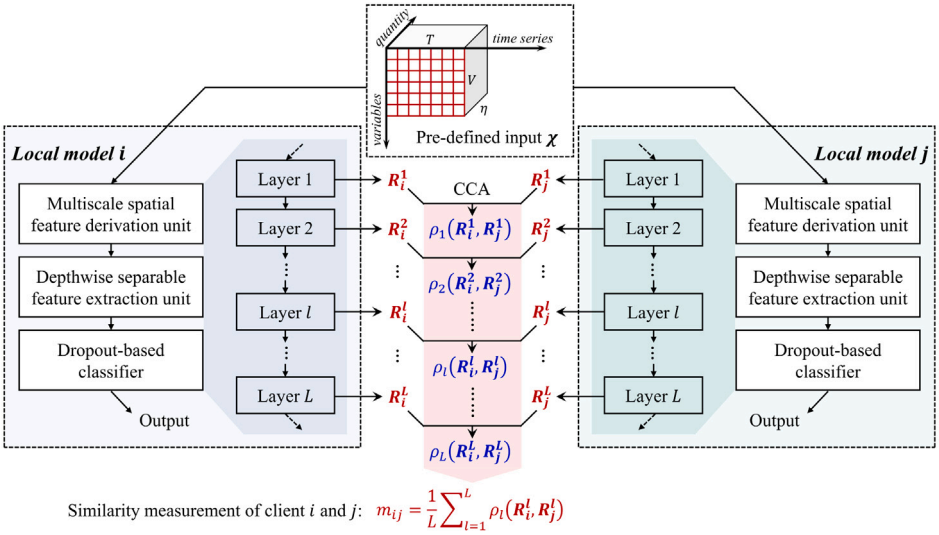


Fig. 10. The schematic diagram illustrating the calculation of model similarity between ith client and jth client.

1. Calculate the degree $\delta_i = \sum_{j=1}^{N} \Delta_{i,j}$ for each local model vertex $\mathcal{M}_i$, and derive the degree matrix Σ for the graph. The unnormalized Laplacian matrix is computed as $\Lambda = \Sigma - \Delta$.
2. Determine the first $\mathcal{K}$ eigenvectors $v_1, v_2, \ldots, v_{\mathcal{K}}$ of Λ, where $\mathcal{K}$ is a hyperparameter representing the number of clusters. Form a matrix $U \in \mathbb{R}^{N\times\mathcal{K}}$ with these eigenvectors as its columns. For $i = 1, 2, \ldots, N$, let $\mu_i \in \mathbb{R}^{\mathcal{K}}$ be the ith row of U.
3. Use the K-means algorithm to cluster $\mu_1, \mu_2, \ldots, \mu_N$, yielding the clusters C.

The number of clusters $\mathcal{K}$ is a hyperparameter that requires determination through experimentation. If the server has no prior knowledge about the clients' characteristics, such as general geographic locations, production capacities, etc., experiment with various settings of $\mathcal{K}$ is necessary to compare the fault detection rates of the cluster models on the test set. The server selects the highest performing $\mathcal{K}$, in terms of fault detection rate, for subsequent clustering. Once $\mathcal{K}$ is determined, the composition of each cluster is also established. Thereafter, the fault diagnosis process does not require re-determination of $\mathcal{K}$.

3.2.3. Cluster model training and model aggregation

Denote the collection of clusters as C. Central server initializes cluster models $\{\mathcal{M}^1, \mathcal{M}^2, \ldots, \mathcal{M}^{|C|}\}$ instead of single global model and transmits parameters of jth cluster model $\theta_{\cdot j}^{(0)}$ to clients who belong to jth cluster. Within jth cluster for any $j = 1, 2, \ldots, |C|$, clients perform updates on their local models and subsequently upload the updated parameters back $\left\{\theta_{1,j}^{(0)}, \theta_{2,j}^{(0)}, \ldots, \theta_{N,j}^{(0)}\right\}$ to the central server.

Cluster model aggregation is then conducted based on federated aggregation algorithms to acquire new cluster model parameters $\theta_{\cdot j}^{(1)}$ for the next communication round. To be general, the aggregation of jth cluster model parameters at communication round t is shown as follows:

$$\theta_{\cdot j}^{(t)} \leftarrow \frac{1}{N}\sum_{i=1}^{N} \theta_{i,j}^{(t-1)}, \forall j \in \{1, 2, \ldots, |C|\} \tag{12}$$

where $\frac{1}{N}\sum_{i=1}^{N} \theta_{i,j}^{(t-1)}$ represents the average aggregation of local trained model parameters. It is noticed that various model aggregation algorithms may differ in their designed approaches and a comprehensive description will be provided subsequently.

Popular aggregation algorithms include FedAvg [19], FedProx [21], and SCAFFOLD [20]. FedAvg is a mean-based approach to simply average the parameters of clients as the details are given in Algorithm B.1. FedProx adds a proximal term to the local loss function. The procedure of FedProx is summarized in Algorithm B.2. SCAFFOLD uses control variates (variance reduction) to decrease the drift of local models and the details are listed in Algorithm B.3.

4. Experiment

In this study, we compare the performance of the baseline models with the proposed clustered federated learning model. The fault diagnosis in this paper includes one normal state and 8 types of faults, thereby yielding in a 9-class classification task. Accuracy, precision, recall, and F1-score are used as performance metrics:

$$\begin{aligned} \text{Accuracy} &= \frac{tp+tn}{tp+tn+fp+fn} \\ \text{Precision} &= \frac{tp}{tp+fp} \\ \text{Recall} &= \frac{tp}{tp+fn} \\ \text{F1-score} &= 2\times\frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{recall}} \end{aligned} \tag{13}$$

where tp, tn, fp and fn represents for true positives, true negatives, false positives and false negatives respectively. In multi-class classification problems, for convenience of evaluation, weighted precision, weighted recall, and weighted F1-score, which are weighted averaged according to the number of samples in each class, are calculated in this paper to evaluate the overall performance.

4.1. Evaluation of LMSRN with traditional federated learning framework

To assess the classification performance and efficiency of the LMSRN model with traditional federated learning framework, we carried out two types of local model comparisons. One set of comparisons involved evaluating the proposed model against baseline models, while the other set focused on conducting ablation experiments for the proposed model.

(1) Baselines: some classic state-of-the-art deep learning models for fault detection are selected to compare with the proposed LMSRN model, The baseline models are described as follows and Table 5 provides the optimized configurations of baseline models:

- CNN [35]: A 4-layer deep network composed of convolutional layer, pooling layer, fully connected layer, and classification layer, originally designed for wind power forecasting by Moayyed et al. (2022). Here, we have changed the model's task to classification and made adaptive adjustments to the model architecture.

Algorithm 1: Clustered Federated Learning for Wind Turbine Fault Diagnosis

Input: Datasets of all wind turbines $\{D_i\}_{i=1}^{N}$, number of communication rounds $\mathcal{R}$, number of local epochs $\mathcal{E}$, local batch size $\mathcal{B}$, learning rate λ, exponential decay rates for the moment estimates $\beta_1, \beta_2 \in [0, 1)$, constant $\epsilon = 10^{-8}$, loss function $\ell(\cdot)$, number of clusters $\mathcal{K}$.

Output: Collection of clusters C, and well-trained cluster model parameters $\theta_{\cdot 1}, \theta_{\cdot 2}, \cdots, \theta_{\cdot |C|}$.

Stage 1. Local LMSRN Model Training
On Clients:
Download initialized global parameters θ from central server as local model parameters $\{\theta_i\}_{i=1}^{N}$
for *each client* $i = 1 : N$ **do**
  Client i updates parameters of local model $\mathcal{M}$ via Adam as shown in Algorithm A.1 with inputs: θ_i, D_i, λ, β_1, β_2, ϵ, $\mathcal{B}$, $\ell_i(\cdot)$
  Upload θ_i to the server
On Server:
Collect each client's local parameters $\{\theta_i\}_{i=1}^{N}$
Stage 2. Representational Canonical Correlation Clustering
On Server:
Acquire representational similarity matrix Δ via Algorithm 2
// Spectral clustering
Construct an undirected weighted graph with vertex set $\{\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_N\}$ and the weighted adjacency matrix is $\Delta = (\Delta_{i,j})_{N \times N}$
Define the degree of a vertex $\mathcal{M}_i$ as $\delta_i = \sum_{j=1}^{N} \Delta_{i,j}$ and form degree matrix Σ
Compute the unnormalized Laplacian matrix $\Lambda = \Sigma - \Delta$
Compute the first $\mathcal{K}$ eigenvectors $v_1, v_2, \cdots, v_{\mathcal{K}}$ of Λ
Let $U \in \mathbb{R}^{N \times \mathcal{K}}$ be the matrix whose columns are $v_1, v_2, \cdots, v_{\mathcal{K}}$
for $i = 1 : N$ **do**
  Define $\mu_i \in \mathbb{R}^{\mathcal{K}}$ as the i-th row of U
Cluster the points $\mu_1, \mu_2, \cdots, \mu_N$ in $\mathbb{R}^{\mathcal{K}}$ using the K-means algorithm to form clusters C
Stage 3. Cluster Model Training
Initialize cluster models $\{\mathcal{M}^1, \mathcal{M}^2, \cdots, \mathcal{M}^{|C|}\}$
for *each cluster* $j = 1 : |C|$ **do**
  Conduct Algorithm B.1, B.2, or B.3 for j-th cluster model training
return *Collection* C *and well-trained cluster model parameters* $\theta_{\cdot 1}, \theta_{\cdot 2}, \cdots, \theta_{\cdot |C|}$

- VGG-19 [36]: A deep network model with a pyramid-like structure formed by using convolutional layers for feature extraction of the same kernel size. VGG is commonly used in computer vision and has been improved for bearing fault diagnosis by Ali Sher et al. (2021).
- MSRAN [17]: Jiang et al. (2022) constructed the MSRAN (multi-scale residual attention network) to simulate local models in the federated learning framework for wind turbine icing detection.

(2) An ablation study is conducted to demonstrate the effectiveness of the depthwise separable feature extraction unit in LMSRN. The study involved replacing this unit with a traditional convolutional neural network block of the same kernel size.

- LMSRN(Conv): change two depthwise separable blocks to commonly-used CNN blocks that consist of one $(1, 1)$ convolution and one $(3, 3)$ or $(5, 5)$ convolution respectively. Make certain that the number of feature maps of CNN block 1 are 32 and that of CNN block 2 are 64, which are the same with LMSRN.

Algorithm 2: Similarity Matrix Computation Based on RCCC

Input: Local parameters of all clients $\{\theta_i\}_{i=1}^{N}$.
Output: Representational similarity matrix Δ.
Initialize a zero matrix Δ
$(\Delta_{i,i}) = 1$
Select weights of convolutional layers and fully connected layers $\{\boldsymbol{R}_1^1, \boldsymbol{R}_1^2, \cdots, \boldsymbol{R}_1^L\}, \{\boldsymbol{R}_2^1, \boldsymbol{R}_2^2, \cdots, \boldsymbol{R}_2^L\}, \cdots, \{\boldsymbol{R}_N^1, \boldsymbol{R}_N^2, \cdots, \boldsymbol{R}_N^L\}$
for *each pair of clients (i, j)* **do**
  for *each layers* $l = 1 : L$ **do**
    Compute $\rho_l(\boldsymbol{R}_i, \boldsymbol{R}_j) = \max_{\omega_i, \omega_j} \frac{\langle \boldsymbol{R}_i\omega_i, \boldsymbol{R}_j\omega_j \rangle}{\|\boldsymbol{R}_i\omega_i\| \cdot \|\boldsymbol{R}_i\omega_i\|}$
  Compute similarity measurement $m_{ij} = \frac{1}{L} \sum_{l=1}^{L} \rho_l(\boldsymbol{R}_i, \boldsymbol{R}_j)$
$(\Delta_{i,j}) = m_{i,j}$, where $i, j = 1, 2, \cdots, N$ and $i \neq j$
return Δ

4.1.1. Settings of LMSRN model with traditional federated learning framework

Table 6 demonstrates detailed configurations of each layer of the LMSRN model. It is important to note that the depthwise layer in the LMSRN's depthwise separable feature extraction unit uses independent convolutions equal to the number of feature maps to extract features. Prior to executing any task with an arbitrary deep learning network, it is crucial to adjust hyperparameters. Specifically, for LMSRN, fine-tuning of hyperparameters such as batch size, local epoch, learning rate, and optimizer is necessary. Hyperparameter tuning for local models is performed by clients using their respective local SCADA datasets. Here, we use the data from WF1 to demonstrate the tuning process.

Initially, we kept the local training epochs constant at 200 and examined the convergence of the LMSRN model during local training as shown in Fig. 11. We plotted the loss curves for different batch sizes as the number of epochs increased, shown in Fig. 11(a), to determine the optimal local epoch for training. Fig. 11(a) highlights that, compared to larger batch sizes, Batch sizes of 32 and 64 result in significant fluctuations in the loss curve, unstable convergence, and a slower descent rate. Larger batch sizes, while leading to more stable convergence, result in slower training. We also monitored the model's generalization performance by recording validation loss from the validation dataset at each iteration round. Fig. 11(b) reveals that for batch sizes of 32 and 64, the model's validation loss displays 'bouncing spikes' with increasing gradient descent steps, indicating potential overfitting. After careful consideration, we selected a batch size of 128 and a local epoch of 100.

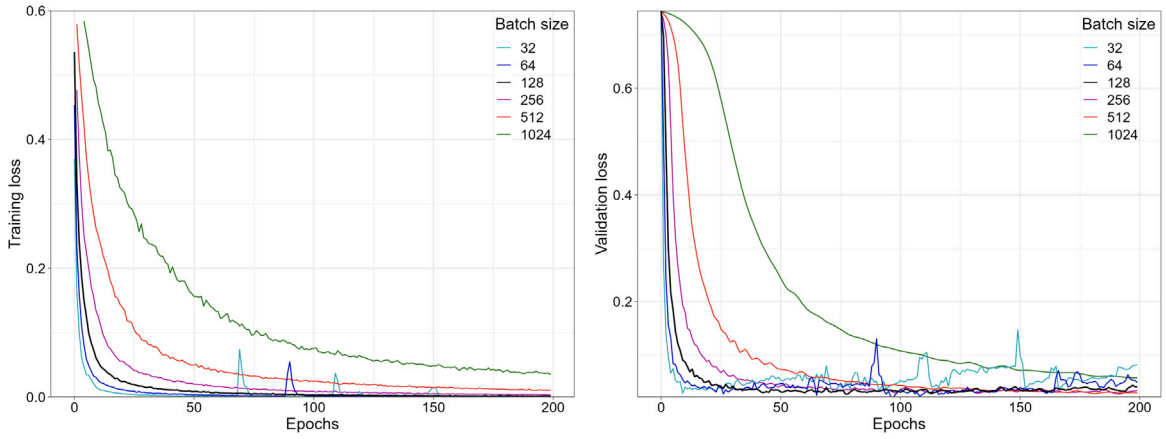
Next, we move on to the selection of the learning rate and optimizer, as illustrated in Fig. 12. We performed several training sessions with identical hyperparameter configurations, generating graphs for training loss. It is observed that a lower learning rate, like 0.0005, requires a greater number of epochs for the model to reach convergence, while a higher learning rate may lead to erratic fluctuations. Considering the various optimizers, we decided to proceed with a learning rate of 0.005 and utilize the *Adam* optimizer.

4.1.2. Comparison of diagnostic performance and computational efficiency

A comparative experiment is designed to showcase the performance and computational efficiency of the proposed LMSRN model. Three model aggregation algorithms – FedAvg, FedProx, and SCAFFOLD – are employed to generate a unified global model. To further validate the effectiveness of the proposed methodology, decentralized and centralized fault diagnosis experiments devoid of the federated learning framework are also conducted. Decentralization signifies that individual clients train their diagnostic models using local datasets in isolation, without aggregating a global model or exchanging model parameters among each other. Centralization, which is in idealized experimental

Table 5
Configuration of the baseline models.

Baseline model	Layer	Filter	Kernel size	Stride	Padding	Activation
CNN	Conv2D	256	(3, 3)	1	Valid	ReLU
	Conv2D	128	(3, 3)	1	Valid	ReLU
	Conv2D	128	(3, 3)	1	Valid	ReLU
	MaxPooling2D		(3, 3)	1	Valid	
	Fully Connected Layer					Sigmoid
	Dropout	0.5				
	Fully Connected Layer					Sigmoid
	Dropout	0.5				
	Fully Connected Layer					Softmax
VGG-19	Conv2D(2)	64	(1, 1), (3, 3)	1	Valid	ReLU
	Conv2D(2)	128	(3, 3), (3, 3)	1	Valid	ReLU
	Conv2D(4)	256	(3, 3), (3, 3), (3, 3), (3, 3)	1	Valid	ReLU
	Conv2D(4)	512	(3, 3), (3, 3), (3, 3), (3, 3)	1	Valid	ReLU
	Conv2D(4)	512	(3, 3), (3, 3), (3, 3), (1, 1)	1	Valid	ReLU
	Fully Connected Layer					Sigmoid
	Dropout	0.5				
	Fully Connected Layer					Softmax
MSRAN	Conv2D	256	(1, 1)	1	Valid	ReLU
	MaxPooling2D		(1, 1)	1	Valid	
	Conv2D Block(4)	32,64	(3, 3), (5, 5), (7, 7), (9, 9)	1	Valid	ReLU
	MaxPooling2D(4)		(3, 3), (5, 5), (7, 7), (9, 9)	1	Valid	
	Self Attention(4)					Hard Sigmoid
	Conv2D	1	(1, 1)	1	Valid	ReLU
	Fully Connected Layer					Sigmoid
	Dropout	0.5				
	Fully Connected Layer					Softmax



(a) Training loss curve of different batch sizes and epochs (b) Validation loss curve of different batch sizes and epochs

Fig. 11. Training and validation loss curves with different hyperparameters (learning rate = 0.001, optimizer is *Adam*).

Table 6
Configuration of the proposed LMSRN model.

Layer	Filter	Kernel size	Stride	Padding	Activation
Conv2D(8)	128	(1, 1), (3, 1), … , (15, 1)	1	Invalid	ReLU
BatchNorm2D(8)					
Pointwise Conv2D	32	(1, 1)	1	Invalid	ReLU
Depthwise Conv2D(32)	1	(3, 3)	1	Valid	ReLU
BatchNorm2D(32)					
Pointwise Conv2D	64	(1, 1)	1	Invalid	ReLU
Depthwise Conv2D(64)	1	(5, 5)	1	Valid	ReLU
BatchNorm2D(64)					
Conv2D	1	(1, 1)	1	Invalid	ReLU
BatchNorm2D					
Fully Connected Layer					Sigmoid
Dropout	0.5				
Fully Connected Layer					Softmax

conditions, assumes that there are no data privacy constraints, allowing clients to upload their data to the server to train a centralized diagnostic model. Subsequently, performance metrics are calculated using the test data from each client. It is important to note that all comparison models underwent hyperparameter tuning by each client as outlined in Section 4.1.1.

The classification result of different models are presented in Table 7. Upon analyzing the precision, recall, and F1-score of different models on the test datasets, it is evident that the proposed LMSRN model surpasses other models in terms of classification performance. Initially, the analysis focuses on comparing the performance of diverse diagnostic models within a decentralized setting. Herein, the proposed LMSRN attains remarkable metrics, with a precision level at 0.90, a recall standing at 0.81, and an F1-score reaching 0.85, thus demonstrating superior outcomes relative to baseline models. It is noteworthy that while the CNN showcases a slightly higher precision score than LMSRN, it exhibits weaker capability in terms of fault recall. Similarly, within a centralized framework, the proposed LMSRN also exhibits diagnostic

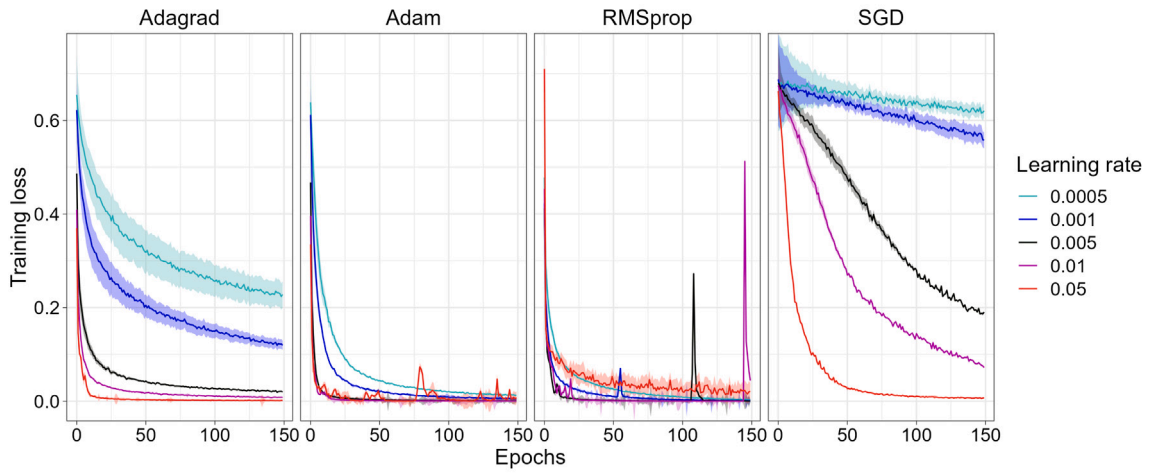


Fig. 12. Training loss curves during model training process of different learning rates and optimizers (batch size = 128).

Table 7
Performance comparison of different diagnostic models.

Model	Framework	Aggregation	Parameters	Model size	Prediction time(s)	Precision	Recall	F1-score
CNN	Centralization		7 951 037	30.34 MB	0.385	0.9819	0.9204	0.9474
	Decentralization				0.383	0.9204	0.7855	0.8362
	Federated	FedAvg			0.386	0.7086	0.5120	0.5741
		FedProx			0.392	0.7086	0.5068	0.5710
		SCAFFOLD			0.391	0.7201	0.5753	0.6237
VGG-19	Centralization		18 655 553	71.24 MB	3.965	0.9854	0.9457	0.9641
	Decentralization				3.963	0.9416	0.2627	0.3932
	Federated	FedAvg			3.956	0.6782	0.5249	0.5546
		FedProx			3.977	0.6864	0.5211	0.5532
		SCAFFOLD			3.962	0.6144	0.5575	0.5559
MSRAN	Centralization		169 483	0.71 MB	2.232	0.9812	0.9517	0.9661
	Decentralization				2.225	0.8756	0.7580	0.8069
	Federated	FedAvg			2.231	0.8881	0.7623	0.7932
		FedProx			2.230	0.8902	0.7742	0.8019
		SCAFFOLD			2.228	0.7962	0.7775	0.7864
LMSRN(Conv)	Centralization		107 852	0.44 MB	0.958	0.9820	0.9502	0.9657
	Decentralization				1.002	0.9080	0.8121	0.8538
	Federated	FedAvg			0.868	0.8055	0.7203	0.7450
		FedProx			0.879	0.8940	0.7610	0.8000
		SCAFFOLD			1.013	0.8678	0.8387	0.8521
LMSRN	Centralization		**47 948**	**0.41 MB**	**1.112**	**0.9859**	**0.9572**	**0.9712**
	Decentralization				**1.180**	**0.9030**	**0.8088**	**0.8451**
	Federated	FedAvg			**1.115**	**0.9108**	**0.7302**	**0.7772**
		FedProx			**1.128**	**0.9124**	**0.7093**	**0.7673**
		SCAFFOLD			**1.329**	**0.8876**	**0.8531**	**0.8691**

performance that surpasses that of the baseline models. Due to the integration of fault samples from all clients, the centralized dataset is larger, thereby resulting in superior diagnostic performance compared to the decentralized framework and federated learning framework. Additionally, the performance of LMSRN(Conv), which integrates traditional convolutional layers, is found to be largely equivalent to that of LMSRN. However, the LMSRN(Conv) requires a substantially larger parameter count, exceeding that of LMSRN by more than a twofold margin.

Subsequently, the contrast centers on the diagnostic performance under the federated learning framework. In this paper, a parallel evaluation of FedAvg, FedProx, and SCAFFOLD is conducted to assess the efficacy of the proposed LMSRN. When considering the aggregation algorithms of FedAvg and FedProx, the precision of LMSRN can exceed 0.91, whereas other models typically fall below 0.90. In scenarios related to real fault diagnosis, recall becomes a critical factor. Models with high recall performance can swiftly detect faults that might otherwise be overlooked, thus preventing losses from sudden wind turbine shutdowns. Upon utilizing SCAFFOLD as the aggregation algorithm for the global LMSRN model, a decrement in precision is noted, with reductions of 2.3% and 2.5% relative to FedAvg and FedProx, respectively. However, the recall value reaches 0.8531, indicating a superior capability in fault detection, a characteristic lacking in baseline models. Similar to the LMSRN, LMSRN(Conv) also exhibits comparably high recall, underscoring the robust fault detection potential inherent in this architecture. In general, the recall values of baseline models are consistently below 0.80, indicating a high rate of missed fault detections. Particularly for VGG-19, which has the largest number of parameters

and significant disparities in parameter distribution, global averaging aggregation leads to a noticeable decline in fault diagnosis performance. It is important to note that, despite the superior classification performance of the proposed LMSRN model compared to other models under traditional federated learning aggregation algorithms, there is still substantial room for improvement overall. This underscores the need for enhancing traditional federated learning frameworks.

From the comparative analysis between decentralization and federated learning framework, it becomes evident that with the SCAFFOLD aggregation algorithm, the LMSRN performs on par with the decentralization, showcasing comparable precision, recall, and F1-score. Global model aggregation through weighted parameter averaging tends to compromise the diagnostic efficacy for minority fault categories, while concurrently augmenting the accuracy for fault types shared extensively among clients. Notably, in scenarios where fault class distributions among certain clients are skewed and data volumes are insufficient, decentralization struggles to attain optimal diagnostic accuracy. A critical distinction lies in the fact that decentralization is inherently incapable of diagnosing fault categories not represented within a client's local dataset. In contrast, federated learning, by amalgamating fault information from all participating clients, confer a broader diagnostic capacity, enabling the diagnosis of a wider range of faults that may not be locally prevalent. Comparing centralization and federated learning framework, the former, owing to its integration of all data, possesses optimal fault diagnosis performance and can thus be considered the upper bound for federated learning algorithms.

Compared to baseline models, the proposed LMSRN model demonstrates notable benefits in relation to both model size and parameters, particularly leading lightweight property by approximately three orders of magnitude when compared to CNN and VGG-19. The number of model parameters critically impacts the speed of forward propagation during prediction. On the same computer setup, the proposed LMSRN model displays faster prediction times when compared to other baseline models. On the ablation study wise, the parameter count of LMSRN(Conv) exceeds that of LMSRN by more than 100%, due to the designed depthwise separable feature extraction unit for lightweighting. Meanwhile, the model size and prediction time of LMSRN are similar to those of LMSRN(Conv). As for model size, the model weight file saves the well-trained model in pytorch dictionary format, resulting in the demand to store more convolutional kernel names when saving the depthwise separable feature extraction unit. In terms of prediction time, frequent splitting and concatenating of feature maps will also slow down the inference speed to some extent.

4.2. Evaluation of LMSRN with clustered federated learning framework

The results of above experiments demonstrate the performance of the proposed LMSRN model in diagnosing faults in wind turbines within a traditional federated learning framework. This section evaluates LMSRN model with clustered federated learning framework. Following a similar approach as before, we will utilize three different model aggregation algorithms within each cluster and compare the proposed framework with various baseline methods and an ablation study to highlight the benefits of clustered federated learning in the context of wind turbine fault diagnosis.

(1) Baseline: Traditional federated learning with one global model, aggregation methods include FedAvg, FedProx, and SCAFFOLD.

(2) Ablation study: Clustered federated learning with other clustering methods.

- Self-organizing map (SOM), an unsupervised learning algorithm, is employed to group clients into clusters [37]. The study will input all trainable model parameters of each local model into the SOM algorithm.
- K-means, a traditional clustering algorithm, divides clients into clusters based on distances to cluster centers [38]. In this research, all trainable parameters of each client's local model will first undergo dimensionality reduction through principal component analysis before being clustered using the K-means algorithm.

Table 8
Details of comparison methods.

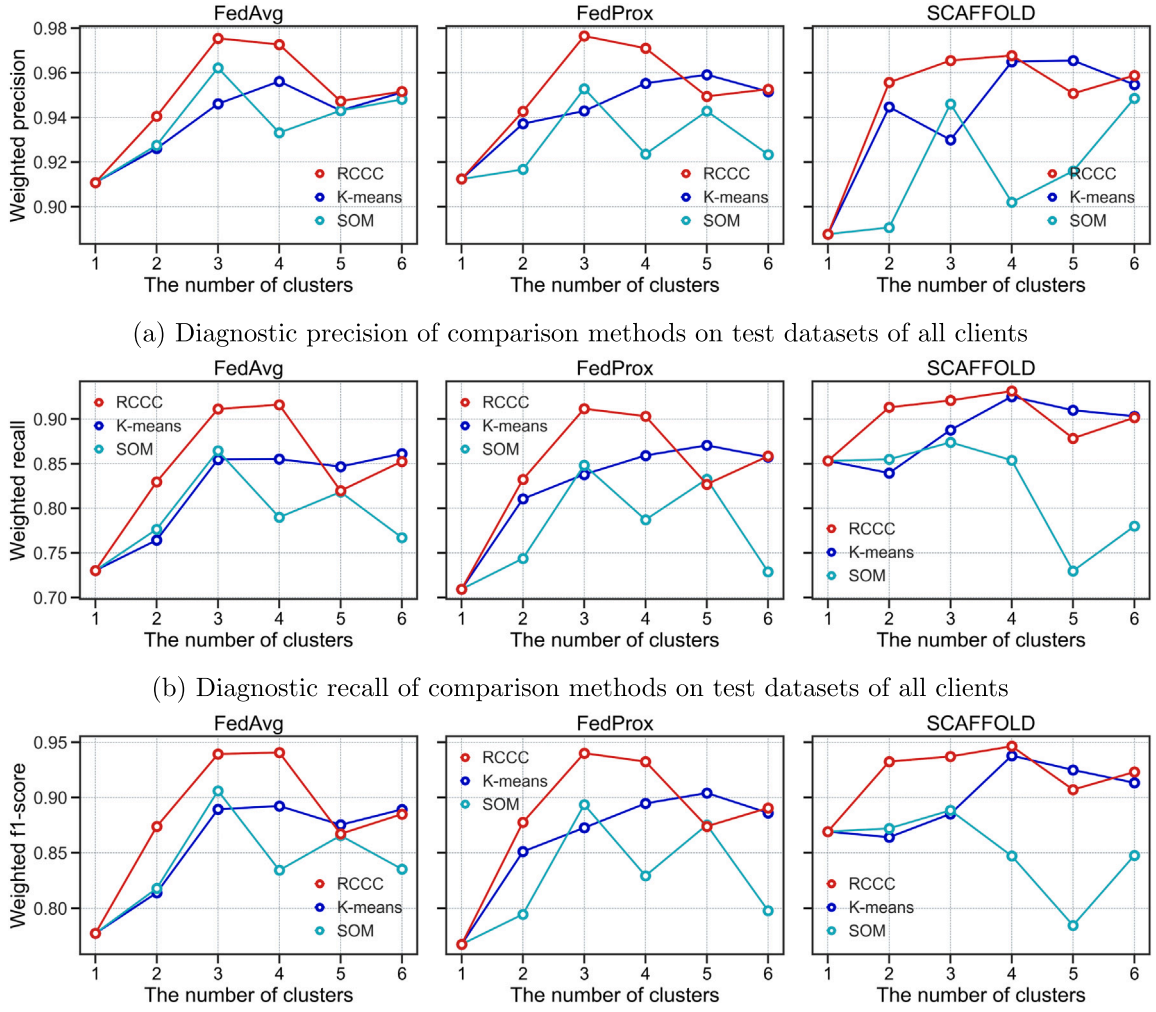
Type	Method	Clustering Algorithm	Aggregation	Number of clusters
Baseline	Traditional FL		FedAvg	1
			FedProx	1
			SCAFFOLD	1
Ablation	Clustered FL	SOM	FedAvg	2, 3, 4, 5, 6
		SOM	FedProx	2, 3, 4, 5, 6
		SOM	SCAFFOLD	2, 3, 4, 5, 6
	Clustered FL	K-means	FedAvg	2, 3, 4, 5, 6
		K-means	FedProx	2, 3, 4, 5, 6
		K-means	SCAFFOLD	2, 3, 4, 5, 6
Proposed	Clustered FL	RCCC	FedAvg	2, 3, 4, 5, 6
		RCCC	FedProx	2, 3, 4, 5, 6
		RCCC	SCAFFOLD	2, 3, 4, 5, 6

The performance of the clustered FL framework is also influenced by clustering algorithms and the number of clusters. In Table 8, we list all the methods for comparison. When the number of cluster is 1, the clustered federated learning method reduces to the traditional FL framework.

4.2.1. Diagnostic performance

Detailed test results of different methods in traditional FL and clustered FL frameworks are presented in Fig. 13. The weighted precision, recall, and F1-score of fault diagnosis across various clustering methods, aggregation algorithms, and the number of clusters are shown in Fig. 13(a), (b), and (c). An analysis of the metric curves indicates an overall increasing trend, highlighting the superior diagnostic performance of clustered FL compared to traditional FL. The precision of fault diagnosis tends to improve with an increase in cluster quantities when RCCC and K-means are used as clustering algorithms, while the SOM algorithm shows less stability. Recall and F1-score are further exhibited in Fig. 13(b) and (c). Notably, the RCCC method consistently outperforms K-means and SOM under both FedAvg, FedProx, and SCAFFOLD aggregation algorithms before the number of clusters reaches 5. The behavior observed at a cluster number of 4 is particularly noticable, as the recall and F1-score curves of RCCC show a turning point. This trend is noticeable for all the three aggregation algorithms. When the number of clusters is set to be 3 or 4, RCCC demonstrates high diagnostic performance, surpassing the other two clustering methods. Additionally, FedProx and SCAFFOLD, derived from FedAvg, help mitigate the slow convergence issue caused by data heterogeneity. The experiments indicate that FedProx and SCAFFOLD are able to effectively address intra-cluster data heterogeneity and achieve remarkable diagnostic capability.

The application of RCCC within the clustered FL framework demonstrates improved fault diagnosis capabilities compared to the traditional FL framework and other clustering methods. Furthermore, experiments have shown that the optimal diagnostic performance is achieved with 3 or 4 clusters. Considering practical scenarios, a lesser number of clusters necessitates fewer cluster models, thereby facilitating management and operational efficiency. Moreover, larger clusters encompass a greater number of clients, aggregating a richer and more diverse repository of fault knowledge. Consequently, this paper selects 3 as the optimal number of clusters. Table 9 show detailed client information of the 3 clusters. Subsequent discussions will delve into the characteristics of these 3 clusters as the RCCC algorithm measures the model similarity between clients. Further analysis of the similarity between different



(a) Diagnostic precision of comparison methods on test datasets of all clients

(b) Diagnostic recall of comparison methods on test datasets of all clients

(c) Diagnostic F1-score of comparison methods on test datasets of all clients

Fig. 13. Comparison of fault diagnosis performance metrics of the clustered FL framework under 3 aggregation algorithms: FedAvg, FedProx, and SCAFFOLD.

wind farms can be conducted based on the similarity matrix Δ output by the RCCC algorithm. By treating each wind farm as a node and the similarity between them as weighted edges, the wind farm network graph illustrated in Fig. 14 can be created. The nodes of the same color in the graph represent wind farms in the same cluster, and the thickness of the edges indicates the level of model similarity between the wind farms. In addition to internal similarities within clusters, there are also edges with relatively small weights connecting different clusters, suggesting a certain degree of similarity between them. This implies the potential presence of shared mechanisms underlying fault occurrence. Remarkably, the resultant clustering bears a significant relationship to both the rated power of the wind turbines and their geographical locations. Specifically, Clusters 1 and 3 are characterized by wind farms housing turbines with a rated capacity of 3.6 MW, diverging from Cluster 2, which includes turbines with a distinctively lower rating of 2 MW. Wind farms grouped within Cluster 1 predominantly occupy positions in Shanghai and Jiangsu, demonstrating spatial adjacency. Likewise, Cluster 3's constituent wind farms are situated in Shanghai and Zhejiang, regions that share a contiguous geographical landscape and are notably proximate to coastal areas, thus potentially influencing the observed clustering patterns. Fig. 15

Table 9
Clustering results of the proposed RCCC approach.

Cluster	Wind farms
Cluster 1	WF5 WF6 WF7 WF8
Cluster 2	WF1 WF2 WF3 WF4 WF12 WF13
Cluster 3	WF9 WF10 WF11

illustrates the confusion matrix using the proposed RCCC approach with 3 clusters. The diagram showcases the diagnostic performance of the cluster models on the test datasets. It is evident that the cluster models perform satisfactorily for most fault categories. Moreover, RCCC shows consistent diagnostic performance across different model aggregation algorithms, highlighting its robustness in diverse aggregating scenarios.

To provide a clearer visualization of the detection rate for fault events as achieved by the proposed method, three-dimensional bar plots are depicted in Fig. 16. The plots document the detection rate for each type of fault across every wind farm, where the detection rate is defined as the ratio of correctly detected faults to the total number of faults of that particular type. Generally, when the cluster number is set

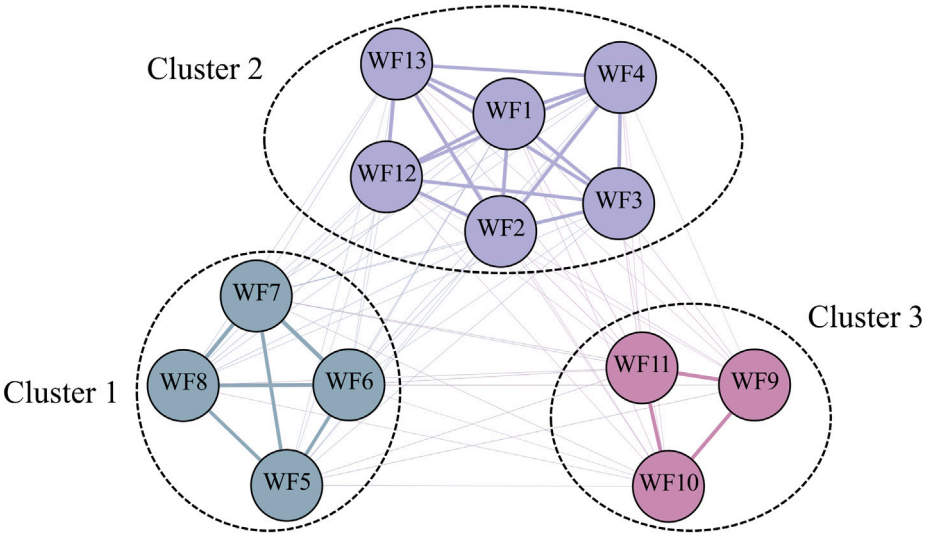


Fig. 14. Network graph depicting clusters of wind farms, where each node represents a unique wind farm. The edges connecting two wind farms show model similarity based on the proposed RCCC, with thicker lines indicating larger similarity.

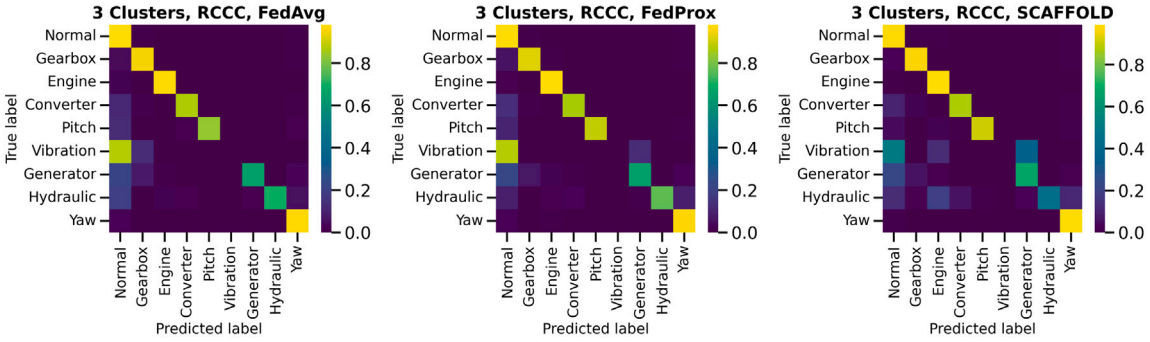
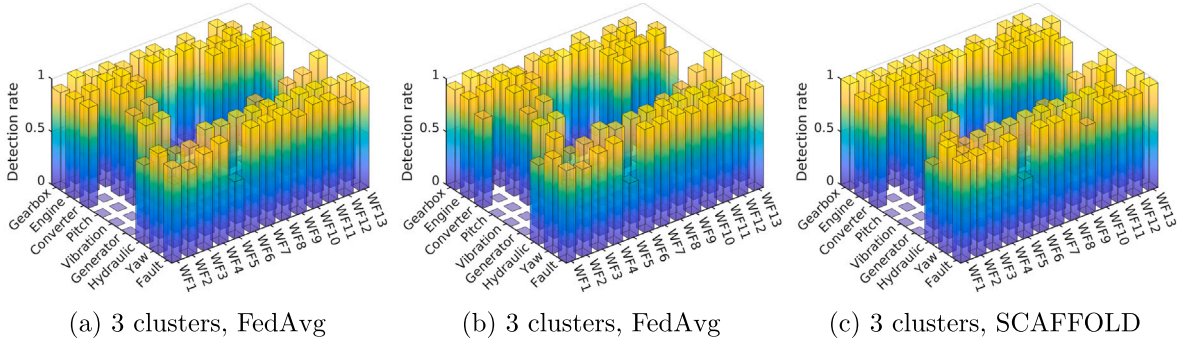


Fig. 15. Confusion matrix of the proposed RCCC approach on test datasets (3 clusters).



(a) 3 clusters, FedAvg (b) 3 clusters, FedAvg (c) 3 clusters, SCAFFOLD

Fig. 16. Detection rate results for 8 distinct types of fault events across 13 wind farms when applying the proposed RCCC algorithm to obtain 3 clusters, where *Fault* in the axis denotes the aggregated detection rate for all fault categories combined. Subfigure (a), (b), and (c) correspondingly represent the results obtained when the model aggregation algorithms employed for the cluster models are FedAvg, FedProx, and SCAFFOLD, respectively.

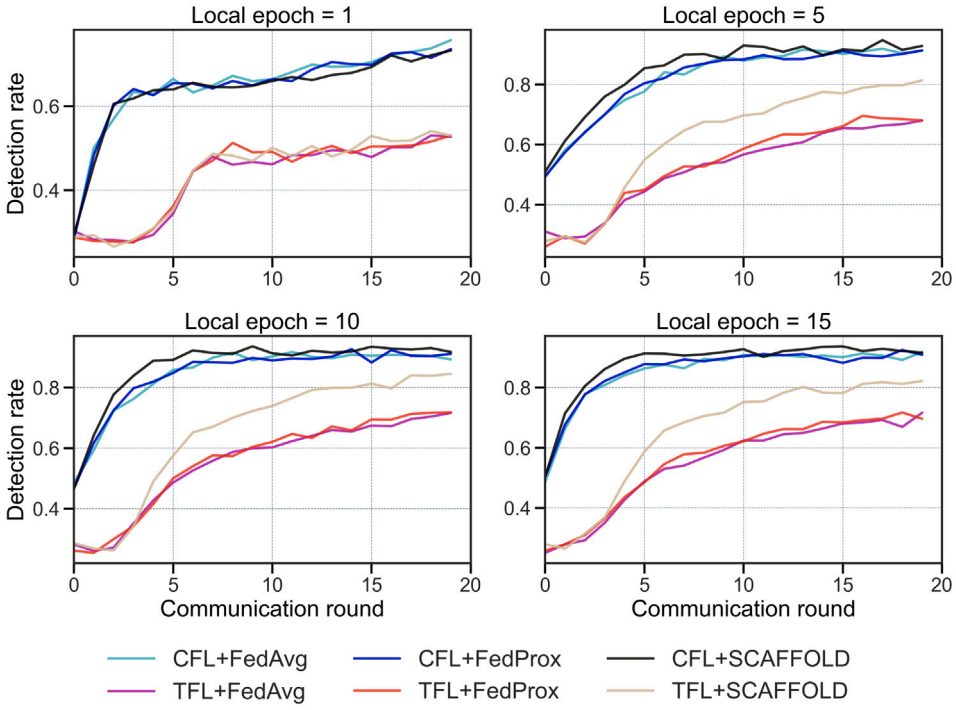


Fig. 17. Overall fault detection rate curves in different local epoch settings. Clustered federated learning (CFL) and traditional federated learning (TFL) are compared under FedAvg, FedProx, and SCAFFOLD.

Table 10
Detection rate of fault events for different methods.

Wind farm	Methods							
	CFL+FedAvg	CFL+FedProx	CFL+SCAFFOLD	TFL+FedAvg	TFL+FedProx	TFL+SCAFFOLD	Decentralization	Centralization
WF1	0.8900	0.8700	0.9400	0.6100	0.6500	0.9300	0.9500	0.9300
WF2	0.8228	0.8268	0.9528	0.5197	0.5984	0.7756	0.7402	0.9528
WF3	0.9100	0.9176	0.9556	0.4575	0.4246	0.6388	0.8200	0.9392
WF4	0.9545	0.9517	0.9766	0.6359	0.6262	0.7766	0.6455	0.9614
WF5	0.5390	0.5228	0.5591	0.4046	0.4288	0.6304	0.7460	0.9449
WF6	0.9876	0.9766	0.9890	0.9245	0.8585	0.9780	0.7747	0.9766
WF7	0.9841	0.9696	0.9797	0.9059	0.8524	0.9711	0.8162	0.9725
WF8	0.9973	0.9973	0.9973	0.9959	0.9932	0.9986	0.9919	0.9986
WF9	0.9232	0.9532	0.8396	0.8062	0.7717	0.8552	0.8875	0.9421
WF10	0.9864	0.9895	0.9895	0.8840	0.8208	0.9172	0.9142	0.9608
WF11	0.9637	0.9521	0.9780	0.7837	0.7772	0.9028	0.8277	0.9469
WF12	0.8702	0.8969	0.9542	0.4466	0.4351	0.8817	0.5840	0.9160
WF13	0.9522	0.9596	0.9890	0.7132	0.7096	0.9706	0.6176	0.9522
Overall	**0.9112**	**0.9115**	**0.9208**	0.7302	0.7093	0.8531	0.8088	0.9572

to 3, all three aggregation algorithms demonstrate a favorable detection rate for faults. Table 10 further illustrates the impact of different training paradigms on the overall fault detection rate when employing LMSRN as the diagnostic model. Among them, CFL (proposed clustered federated learning) exhibits superior performance, achieving an overall detection rate greater than 0.9 across all wind farms, which outperforms both TFL (traditional federated learning) and Decentralization methods. Furthermore, the diagnostic performance of CFL favorably approaches that achieved under the idealized centralized framework. This comparative analysis reaffirms the superiority and innovation of the proposed methodology in facilitating collaborative fault diagnosis among multiple wind farms while preserving data privacy.

4.2.2. Communication cost

In federated learning, communication cost arises from two primary aspects: the scale of model parameter transmission and the frequency of model broadcasting and aggregating. The scale of parameter transmission has been thoroughly addressed in Section 4.1, where a lightweight and efficient model, LMSRN, is selected as the diagnostic model for parameter transfer between clients and the server. By employing LMSRN, the volume of data transmitted during each round of communication is significantly reduced, contributing to lower overall communication costs. The frequency of model broadcasting and aggregating is directly reflected in the number of communication rounds, which is global epoch $\mathcal{R}$. When the number of local epochs $\mathcal{E}$ is fixed, there exists an optimal number of communication rounds $\mathcal{R}$ that enables

the models in federated learning to reach convergence [39]. Therefore, assessing the communication cost of model aggregation algorithms can be done by examining the number of communication rounds required for convergence. A lower frequency of global communication close to convergence indicates a lower communication cost. To substantiate the superior communication efficiency of the proposed clustered federated learning (CFL) framework, comparative experiments are conducted by setting the cluster number to 3 and employing the LMSRN as the diagnostic model. These experiments are specifically designed to contrast the convergence velocity of CFL against that of traditional federated learning (TFL). As illustrated in Fig. 17, under various settings of local epochs, the proposed CFL consistently exhibits rapid and robust convergence, achieving high detection rate levels. Among different aggregation algorithms, SCAFFOLD notably accelerates convergence in CFL framework, making it particularly suitable for scenarios characterized by data heterogeneity across multiple wind farms.

5. Conclusion

This paper introduces a clustered federated learning framework for collaborative fault diagnosis of multiple wind turbines in various wind farms. The framework trains server-side global models for fault diagnosis while maintaining the privacy of local wind turbine client data. By grouping wind farms with similar models into clusters, the proposed framework addresses data heterogeneity issues and generates clustered models. Similarity between parameterized deep fault diagnosis models is measured using canonical correlation coefficients, and clusters are obtained through representational canonical correlation clustering (RCCC). To reduce communication overhead and speed up fault diagnosis, a lightweight multiscale separable residual network (LMSRN) is introduced as the wind turbine fault diagnosis model. The LMSRN utilizes multiscale spatial feature derivation units to extract correlations among wind turbine variables and depthwise separable feature extraction units to reduce the model's parameter quantity.

The study presents experimental results comparing the proposed LMSRN model and clustered federated learning framework with baseline methods. The findings show that the LMSRN model outperforms other baseline models in classification performance and lightweight properties. Additionally, the clustered federated learning framework with multiple cluster models shows better fault diagnosis performance and more efficient convergence compared to the traditional federated learning framework with a single global model. The study concludes that the RCCC algorithm efficiently clusters wind farms into groups, with optimal diagnostic performance ability observed when using 3 clusters.

Certain constraints of the algorithms and framework are acknowledged, such as the need to determine the optimal number of client clusters dynamically and improve communication between clusters for enhanced fault pre-diagnostic capability. Furthermore, future research should explore tailored federated learning algorithms to address class imbalance effectively.

CRediT authorship contribution statement

Rui Zhou: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation. **Yanting Li:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Data curation, Conceptualization. **Xinhua Lin:** Resources, Software.

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.

Data availability

Data will be made available on request.

Acknowledgments

The authors would like to acknowledge National Natural Science Foundation of China (General Program) 72072114 for funding the research. This work was also partially supported by SJTU Kunpeng& Ascend Center of Excellence.

Appendix A. Local model parameter update

The details of the local model parameter update algorithm are provided below in Algorithm A.1.

Algorithm A.1: Adaptive Moment Estimation (Adam)

Input: θ_i, D_i, λ, β_1, β_2, ϵ, $\mathcal{B}$, $\ell_i(\cdot)$.
Output: θ_i.

$s \leftarrow 0$ Initialize 1st moment vector
$r \leftarrow 0$ Initialize 2nd moment vector
$t \leftarrow 0$ Initialize time step
while $t < \frac{|D_i|}{\mathcal{B}}$ **do**
  Get gradient $g \leftarrow \nabla_{\theta_i} \ell_i(\theta_i)$
  Update biased 1st moment estimate $s \leftarrow \beta_1 s + (1-\beta_1) g$
  Update biased 2nd raw moment estimate $r \leftarrow \beta_2 r + (1-\beta_2) g^2$
  Correct bias in 1st moment $\hat{s} \leftarrow \frac{s}{1-\beta_1^t}$
  Correct bias in 2nd moment $\hat{r} \leftarrow \frac{r}{1-\beta_2^t}$
  Update local parameters $\theta_i \leftarrow \theta_i - \lambda \frac{\hat{s}}{\sqrt{\hat{r}}+\epsilon}$
  $t \leftarrow t+1$
return θ_i.

Appendix B. Model aggregation algorithms

Details of the three model aggregation algorithms mentioned in Section 3 are provided below.

Algorithm B.1: Federated Averaging (FedAvg)

Input: j-th cluster $C \in \mathcal{C}$ and clients belong to it, datasets of wind turbines $\left\{D_i^j\right\}_{i=1}^{|C|}$ in cluster C, number of communication rounds $\mathcal{R}$, number of local epochs $\mathcal{E}$, local batch size $\mathcal{B}$, learning rate λ, exponential decay rates for the moment estimates $\beta_1, \beta_2 \in [0,1)$, constant $\epsilon = 10^{-8}$, loss function $\ell(\cdot)$.
Output: Well-trained j-cluster model parameters $\theta_{\cdot j}$.

On Server:
Initialize cluster model parameters $\theta_{\cdot j}^{(0)}$
for *each communication round* $t = 1 : \mathcal{R}$ **do**
  for *each client* $i = 1 : |C|$ **do**
    $\theta_{i,j}^{(t-1)} \leftarrow LocalUpdate\left(D_i^j, \theta_{\cdot j}^{(t-1)}\right)$
  $\theta_{\cdot j}^{(t)} \leftarrow \frac{1}{|C|} \sum_{i \in C} \theta_{i,j}^{(t-1)}$
LocalUpdate (D_i, θ_i):
for *each local epoch* $e = 1 : \mathcal{E}$ **do**
  θ_i is updated via Adam shown in Algorithm A.1 with inputs: θ_i, D_i, λ, β_1, β_2, ϵ, $\mathcal{B}$, $\ell_i(\cdot)$

Algorithm B.2: Federated Proximal (FedProx)

Input: j-th cluster $C \in \mathcal{C}$ and clients belong to it, datasets of wind turbines $\left\{D_i^j\right\}_{i=1}^{|C|}$ in cluster C, number of communication rounds $\mathcal{R}$, number of local epochs $\mathcal{E}$, local batch size $\mathcal{B}$, learning rate λ, exponential decay rates for the moment estimates $\beta_1, \beta_2 \in [0,1)$, constant $\epsilon = 10^{-8}$, loss function $\ell(\cdot)$.

Output: Well-trained j-cluster model parameters $\theta_{\cdot j}$.

On Server:
Initialize cluster model parameters $\theta_{\cdot j}^{(0)}$
for *each communication round* $t = 1 : \mathcal{R}$ **do**
  for *each client* $i = 1 : |C|$ **do**
    $\theta_{i,j}^{(t-1)} \leftarrow LocalUpdate\left(D_i^j, \theta_{\cdot j}^{(t-1)}\right)$
    $\theta_{i,j}^{(t-1)} \leftarrow \theta_{i,j}^{(t-1)} + \lambda\left(\theta_{i,j}^{(t-1)} - \theta_{\cdot j}^{(t-1)}\right)$
  $\theta_{\cdot j}^{(t)} \leftarrow \frac{1}{|C|}\sum_{i \in C} \theta_{i,j}^{(t-1)}$
$LocalUpdate\left(D_i, \theta_i\right)$:
for *each local epoch* $e = 1 : \mathcal{E}$ **do**
  θ_i is updated via Adam shown in Algorithm A.1 with inputs: θ_i, D_i, λ, β_1, β_2, ϵ, $\mathcal{B}$, $\ell_i(\cdot)$

Algorithm B.3: Stochastic Controlled Averaging (SCAFFOLD)

Input: j-th cluster $C \in \mathcal{C}$ and clients belong to it, datasets of wind turbines $\left\{D_i^j\right\}_{i=1}^{|C|}$ in cluster C, number of communication rounds $\mathcal{R}$, number of local epochs $\mathcal{E}$, local batch size $\mathcal{B}$, learning rate λ, exponential decay rates for the moment estimates $\beta_1, \beta_2 \in [0,1)$, constant $\epsilon = 10^{-8}$, loss function $\ell(\cdot)$.

Output: Well-trained j-cluster model parameters $\theta_{\cdot j}$.

On Server:
Initialize cluster model parameters $\theta_{\cdot j}^{(0)}$ and control variates $\boldsymbol{c}$
Communicate $\left(\theta_{\cdot j}^{(0)}, \boldsymbol{c}\right)$ to clients
for *each communication round* $t = 1 : \mathcal{R}$ **do**
  for *each client* $i = 1 : |C|$ **do**
    Initialize control variates $\boldsymbol{c_i} \leftarrow \boldsymbol{c}$
    $\theta_{i,j}^{(t-1)} \leftarrow LocalUpdate\left(D_i^j, \theta_{\cdot j}^{(t-1)}, \boldsymbol{c}, \boldsymbol{c_i}\right)$
    $\boldsymbol{c_i^+} \leftarrow \boldsymbol{c_i} - \boldsymbol{c} + \frac{\lambda}{\mathcal{E}}\left(\theta_{\cdot j}^{(t-1)} - \theta_{i,j}^{(t-1)}\right)$
    $\Delta \boldsymbol{c_i} \leftarrow \boldsymbol{c_i^+} - \boldsymbol{c_i}$
    $\boldsymbol{c_i} \leftarrow \boldsymbol{c_i^+}$
  $\theta_{\cdot j}^{(t)} \leftarrow \frac{1}{|C|}\sum_{i \in C} \theta_{i,j}^{(t-1)}$
  $\boldsymbol{c} \leftarrow \boldsymbol{c} + \frac{1}{|C|}\sum_{i \in C} \Delta \boldsymbol{c_i}$
$LocalUpdate\left(D_i, \theta_i, \boldsymbol{c}, \boldsymbol{c_i}\right)$:
for *each local epoch* $e = 1 : \mathcal{E}$ **do**
  θ_i is updated via Adam shown in Algorithm A.1 with inputs: θ_i, D_i, λ, β_1, β_2, ϵ, $\mathcal{B}$, $\ell_i(\cdot)$
  $\theta_i \leftarrow \theta_i - \lambda\left(\boldsymbol{c} - \boldsymbol{c_i}\right)$

References

[1] Council GWE. Global Wind Report 2023. 2023.
[2] International Energy Agency. Wind electricity. 2023.
[3] Stehly T, Duffy P. 2021 Cost of wind energy review. 2022.
[4] Zhao Y, Li D, Dong A, Lin J, Kang D, Shang L. Fault prognosis of wind turbine generator using SCADA data. In: 2016 North American power symposium. NAPS, 2016, p. 1–6.
[5] Yin S, Wang G, Karimi HR. Data-driven design of robust fault detection system for wind turbines. Mechatronics 2014;24(4):298–306.
[6] Zou L, Li Y, Xu F. An adversarial denoising convolutional neural network for fault diagnosis of rotating machinery under noisy environment and limited sample size case. Neurocomputing 2020;407:105–20.
[7] Helbing G, Ritter M. Deep learning for fault detection in wind turbines. Renew Sustain Energy Rev 2018;98:189–98.
[8] Jia F, Lei Y, Lin J, Zhou X, Lu N. Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data. Mech Syst Signal Process 2016;72–73:303–15.
[9] Bach-Andersen M, Rømer-Odgaard B, Winther O. Deep learning for automated drivetrain fault detection. Wind Energy 2018;21(1):29–41.
[10] Liu H, Zhou J, Zheng Y, Jiang W, Zhang Y. Fault diagnosis of rolling bearings with recurrent neural network-based autoencoders. ISA Trans 2018;77:167–78.
[11] Tong Z, Li W, Zhang B, Jiang F, Zhou G. Bearing fault diagnosis under variable working conditions based on domain adaptation using feature transfer learning. IEEE Access 2018;6:76187–97.
[12] Yang B, Lei Y, Jia F, Xing S. An intelligent fault diagnosis approach based on transfer learning from laboratory bearings to locomotive bearings. Mech Syst Signal Process 2019;122:692–706.
[13] Chen W, Qiu Y, Feng Y, Li Y, Kusiak A. Diagnosis of wind turbine faults with transfer learning algorithms. Renew Energy 2021;163:2053–67.
[14] Li Y, Jiang W, Zhang G, Shu L. Wind turbine fault diagnosis based on transfer learning and convolutional autoencoder with small-scale data. Renew Energy 2021;171:103–15.
[15] Zhang G, Li Y, Jiang W, Shu L. A fault diagnosis method for wind turbines with limited labeled data based on balanced joint adaptive network. Neurocomputing 2022;481:133–53.
[16] Yang Q, Liu Y, Chen T, Tong Y. Federated machine learning: Concept and applications. ACM Trans Intell Syst Technol 2019;10(2).
[17] Jiang G, Fan W, Li W, Wang L, He Q, Xie P, et al. DeepFedWT: A federated deep learning framework for fault detection of wind turbines. Measurement 2022;199:111529.
[18] Cheng X, Shi F, Liu Y, Liu X, Huang L. Wind turbine blade icing detection: a federated learning approach. Energy 2022;254:124441.
[19] McMahan B, Moore E, Ramage D, Hampson S, Arcas BAy. Communication-efficient learning of deep networks from decentralized data. In: Singh A, Zhu J, editors. Proceedings of the 20th international conference on artificial intelligence and statistics. Proceedings of machine learning research, Vol. 54, PMLR; 2017, p. 1273–82.
[20] Karimireddy SP, Kale S, Mohri M, Reddi S, Stich S, Suresh AT. Scaffold: Stochastic controlled averaging for federated learning. In: International conference on machine learning. PMLR; 2020, p. 5132–43.
[21] Li T, Sahu AK, Zaheer M, Sanjabi M, Talwalkar A, Smith V. Federated optimization in heterogeneous networks. In: Proceedings of machine learning and systems. vol. 2, 2020, p. 429–50.
[22] Briggs C, Fan Z, Andras P. Federated learning with hierarchical clustering of local updates to improve training on non-IID data. In: 2020 international joint conference on neural networks. IJCNN, 2020, p. 1–9.
[23] Palihawadana C, Wiratunga N, Wijekoon A, Kalutarage H. FedSim: Similarity guided model aggregation for federated learning. Neurocomputing 2022;483:432–45.
[24] Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: Synthetic minority over-sampling technique. J Artif Int Res 2002;16(1):321–57.
[25] Zhang K, Chen J, Zhang T, Zhou Z. A compact convolutional neural network augmented with multiscale feature extraction of acquired monitoring data for mechanical intelligent fault diagnosis. J Manuf Syst 2020;55:273–84.
[26] Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. In: International conference on artificial intelligence and statistics. 2011.
[27] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C. MobileNetV2: Inverted residuals and linear bottlenecks. In: 2018 IEEE/CVF conference on computer vision and pattern recognition. 2018, p. 4510–20.
[28] Zhu X, Wang R, Fan Z, Xia D, Liu Z, Li Z. Gearbox fault identification based on lightweight multivariate multidirectional induction network. Measurement 2022;193:110977.
[29] Chollet F. Xception: Deep learning with depthwise separable convolutions. In: 2017 IEEE conference on computer vision and pattern recognition. CVPR, 2017, p. 1800–7.
[30] Morcos A, Raghu M, Bengio S. Insights on representational similarity in neural networks with canonical correlation. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R, editors. Advances in neural information processing systems. Vol. 31, Curran Associates, Inc.; 2018.
[31] Hotelling H. The most predictable criterion. J Educ Psychol 1935;26:139–42.
[32] Kornblith S, Norouzi M, Lee H, Hinton G. Similarity of neural network representations revisited. In: Chaudhuri K, Salakhutdinov R, editors. Proceedings of the 36th international conference on machine learning. Proceedings of machine learning research, Vol. 97, PMLR; 2019, p. 3519–29.
[33] Raghu M, Gilmer J, Yosinski J, Sohl-Dickstein J. SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In: Proceedings of the 31st international conference on neural information processing systems. NIPS '17, Red Hook, NY, USA: Curran Associates Inc.; 2017, p. 6078–87.

[34] Duong L, Zhou J, Nassar J, Berman J, Olieslagers J, Williams AH. Representational dissimilarity metric spaces for stochastic neural networks. 2022, arXiv arXiv:2211.11665.
[35] Moayyed H, Moradzadeh A, Mohammadi-Ivatloo B, Aguiar AP, Ghorbani R. A cyber-secure generalized supermodel for wind power forecasting based on deep federated learning and image processing. Energy Convers Manage 2022;267:115852.
[36] Ali Sher M, Muhammad U, Yu X, Hu Q. Fault diagnosis of rolling element bearing using a mesh of continuous wavelet transform and visual geometry group 19 (VGG-19). In: 2021 IEEE international conference on artificial intelligence and computer applications. ICAICA, 2021, p. 102–6.
[37] Kohonen T. The self-organizing map. Proc IEEE 1990;78(9):1464–80.
[38] Jain AK, Dubes RC. Algorithms for clustering data. Prentice-Hall, Inc.; 1988.
[39] Li X, Huang K, Yang W, Wang S, Zhang Z. On the convergence of FedAvg on non-IID data. In: International conference on learning representations. 2020.