mathematics

MDPI

Article

FedGCN: Federated Learning-Based Graph Convolutional Networks for Non-Euclidean Spatial Data

Kai Hu [1,2,*], Jiasheng Wu [1,3], Yaogen Li [1,2], Meixia Lu [1,2], Liguo Weng [1,2] and Min Xia [1,2]

[1] School of Automation, Nanjing University of Information Science and Technology, Nanjing 210044, China; 20181223081@nuist.edu.cn (J.W.); 20201249100@nuist.edu.cn (Y.L.); 20191223036@nuist.edu.cn (M.L.); 002311@nuist.edu.cn (L.W.); xiamin@nuist.edu.cn (M.X.)
[2] Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET), Nanjing University of Information Science and Technology, Nanjing 210044, China
[3] School of Economics and Business Management, Nanjing University of Science and Technology, Nanjing 210094, China
* Correspondence: 001600@nuist.edu.cn; Tel.: +86-137-7056-9871

Abstract: Federated Learning (FL) can combine multiple clients for training and keep client data local, which is a good way to protect data privacy. There are many excellent FL algorithms. However, most of these can only process data with regular structures, such as images and videos. They cannot process non-Euclidean spatial data, that is, irregular data. To address this problem, we propose a Federated Learning-Based Graph Convolutional Network (FedGCN). First, we propose a Graph Convolutional Network (GCN) as a local model of FL. Based on the classical graph convolutional neural network, TopK pooling layers and full connection layers are added to this model to improve the feature extraction ability. Furthermore, to prevent pooling layers from losing information, cross-layer fusion is used in the GCN, giving FL an excellent ability to process non-Euclidean spatial data. Second, in this paper, a federated aggregation algorithm based on an online adjustable attention mechanism is proposed. The trainable parameter ρ is introduced into the attention mechanism. The aggregation method assigns the corresponding attention coefficient to each local model, which reduces the damage caused by the inefficient local model parameters to the global model and improves the fault tolerance and accuracy of the FL algorithm. Finally, we conduct experiments on six non-Euclidean spatial datasets to verify that the proposed algorithm not only has good accuracy but also has a certain degree of generality. The proposed algorithm can also perform well in different graph neural networks.

Keywords: federated learning; graph convolutional neural network; non-Euclidean spatial data; attention mechanism

MSC: 68T07

Citation: Hu, K.; Wu, J.; Li, Y.; Lu, M.; Wang, L.; Xia, M. FedGCN: Federated Learning-Based Graph Convolutional Networks for Non-Euclidean Spatial Data. *Mathematics* **2022**, *10*, 1000. https://doi.org/10.3390/math10061000

Academic Editor: Ezequiel Lopez Rubio

Received: 13 January 2022
Accepted: 16 March 2022
Published: 21 March 2022

1. Introduction

Federated learning [1–3] is a particular type of distributed machine learning. Figure 1 shows the framework of federated learning. FL enables multiple clients to perform joint training under the premise of ensuring that data are not shared. Only local model training parameters are shared in the training process. A client is no longer a single individual, and the client can thoroughly learn the training experience from others. This approach provides a possible solution to data island problems. McMahan et al. [4] proposed the Federated Average (FedAvg) algorithm. They were the first to propose a relatively complete federated learning framework based on deep learning. The framework takes full account of the common Non-Independently Identically Distributed (Non-IID) data and communications costs in FL. FedAvg evaluates its performance on five different models and four datasets. The combination of FL [5–7] and Deep Learning [8–12] has opened a new world of research.

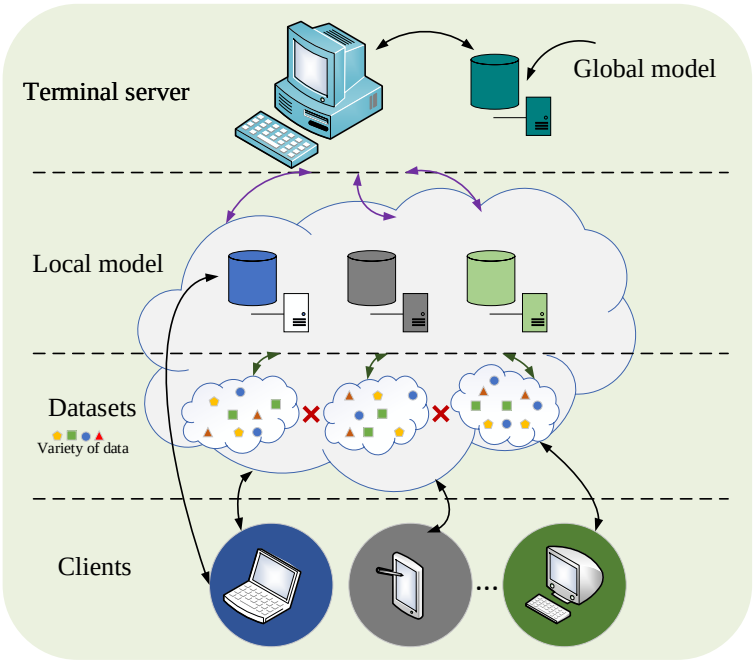


Figure 1. Algorithm framework of federated learning.

At present, many variants of FedAvg and even new FL frameworks appeared to solve various challenges faced by federated learning. Yang et al. [13] proposed a logical regression method of center-based vertical federated learning, which solved the logical regression in vertical federated learning. Hartmann et al. [14] proposed a federated support vector machine in 2019, which optimizes and protects parameters by updating blocks of local modules and hashing attributes. Liu et al. [15] proposed a federated decision tree model, which uses local participants to transmit the ranking of model parameters to replace the original federated learning process of constantly uploading model parameters. Peng et al. [16] proposed an Unsupervised Federated learning Domain Adaptation (UFDA) method, which solves the phenomenon of negative transfer caused by domain irrelevance. Yurochkin et al. [17] proposed to apply a Bayesian network to federated learning, and they developed and studied a probabilistic federated learning framework, with special emphasis on training an aggregation neural network model, matching estimated local model parameters between data sources, in order to build a global network and improve the accuracy of the aggregation model in each client. Nadiger et al. [18] proposed federated reinforcement learning and proposed a federated reinforcement technology aimed at a long personalization time. Its main goal is to improve the personalization time. Fei Chen et al. [19] proposed a federated meta-learning framework, known as Federated Meta-learning (FedMeta). Experiments on the same dataset show that the communication cost of FedMeta is 2.82–4.33 times lower than that of FedAvg, and its convergence speed is faster. Compared with FedAvg, which uses the same number of local training iterations for all clients in each round of global model updates, FedProx [20] allows different local training iterations in different clients according to the available system resources of the client, mainly to solve the problem of heterogeneity in federated learning that FedAvg has not dealt with, and provides stronger robustness. The datasets used in these excellent federated learning algorithms are commonly used in Euclidean space and can process daily data such as image, video, and speech recognition [21,22].

However, in real life, there are a large number of data in non-Euclidean space that contain complex interdependent graph nodes, which cannot be effectively processed by conventional federated learning and deep learning. Still, there are many non-Euclidean spatial data in real life, and these data contain complex graph node relationships that cannot be effectively handled by conventional federated learning and deep learning. Therefore,

GCNs, which can process non-Euclidean spatial data, have become another hot research direction in artificial intelligence. The GCN proposed by Kipf [23] is the semi-supervised learning of graph structure data. The method does not require all nodes to have corresponding labels for node tasks. Hamilton [24] et al. proposed an inductive GCN (Graph Sample and Aggregate, GraphSage) that can be applied to dynamic graph data. GraphSage is not the embedded training for each node alone. However, it aggregates all the samplings from each node's neighbors of a certain number (for example, calculating the mean of all sampled neighborhood eigenvectors). Unlike the graph convolution based on the spatial domain of GraphSage, the GCN proposed by Kipf et al. uses the first-order approximation of ChebNet [25] to determine the convolution structure. Veli Kovi et al. [26] applied the attention mechanism to graph convolutional layers. The attention mechanism allows the model to process input information of different scales, focusing on the most relevant part of the input information. After experiments, these Graph Neural Network (GNN) algorithms achieved excellent performance in many tasks, such as node classification, edge prediction, and graph classification. However, the design of a GNN is mainly based on empirical intuition, heuristics, or experimental trials. Although it shows an excellent non-Euclidean spatial data processing ability, there is no corresponding theoretical understanding of the nature and limitations of GNNs. Xu [27] proposed a theoretical framework for analyzing the powerful expressive ability of GNNs and described the performance of different GNN variants in extracting features. Di [28] proposed a multi-graph convolutional neural network model to predict the traffic at a station and observe the bicycle sharing system from the graph's perspective. To endow the graph neural network with privacy security, Mei [29] used the federated average algorithm to train the improved graph neural network, which hides the specific information of nodes. Zhang [30] added attention to the spatio-temporal graph neural network. They also added differential privacy protection to the adjacency matrix of the graph so as to train through the federated average algorithm.

In order to address some of the above problems, this paper proposes a Federated Learning Framework-Based Graph Convolutional Network (FedGCN). The framework uses a graph convolutional neural network to replace the deep convolutional neural network used in the classical FL framework. Moreover, FedGCN aggregates local model parameters using the attention mechanism. Overall, this paper makes the following contributions.

(1) The FL capabilities in processing non-Euclidean spatial data are enhanced. First, based on the ability of the GCN to process non-Euclidean spatial data, this paper proposes a GCN to build local models for federated learning clients. The difference between the GCN proposed in this paper and the classical GCN is the fact that TopK graph pooling layers and full connection layers are added to extract node features. Afterwards, to prevent the loss of important information, a cross-layer fusion mechanism is adopted to fuse the features extracted from all layers.

(2) Then, to prevent the loss of important information, a cross-layer fusion mechanism is adopted to fuse the features extracted from all layers.

(3) The fault tolerance of FL aggregation and the accuracy of the global model are improved. Each local model is assigned the same weight according to the average aggregation algorithm. In this case, the aggregated global model cannot absorb the local model's effective training experience well. This paper proposes a federated aggregation algorithm based on an attention mechanism that can be adjusted online. In order to make better use of the GCN in Contribution 1 for distributed training, this paper uses the federated learning mechanism to jointly train several local GCN models, so that the federated learning can not only work in the regular data, but also obtain a global model with a high generalization performance by learning non-Euclidean spatial data.

(4) In order to finely process the local model parameters, including the errors contained in the data itself, an attention mechanism is introduced to replace the previous average mechanism to obtain federated weights. Different from the attention mechanism with fixed parameters, this paper uses a trainable parameter ρ to continuously adjust the aggregation weight. This trainable parameter ρ is trained by a simple perceptron. It provides appropriate

attention weights for each client's model parameters, and these weights are applied to parameters in every layer so as to accurately obtain the optimal global model using the training parameters of each local model.

The overall framework of this paper is as follows. The first section introduces the research status of relevant technologies. The second section briefly introduces the classical federated average learning algorithm and GCNs. In the third section, federated learning based on a graph convolutional neural network is presented in detail. The fourth section verifies the proposed algorithm's effectiveness through experiments on six non-Euclidean spatial datasets. The fifth section is the conclusion of the paper.

To some extent, the processing of graph data, i.e., the traditional GNN, suffers from certain defects. For example, the undersampling of graph data has the problem of insufficient feature extraction. Junhyun Lee et al. [31,32] proposed a graph pooling method based on self-attention. The Self-Attention Graph Pooling (SAGPool) of graph convolution allows the pooling method to consider two node features and graph topology at the same time. Its structure features hierarchical pooling, a consideration of node features and a graph topology, reasonable complexity, and end-to-end learning. In federated learning, graph data also need to be processed. Shaoxiong Ji et al. [33] showed that a novel layer-by-layer attention joint optimization can measure the importance of selected content for modeling clients and speed up the learning process. The proposed attention aggregation method minimizes server use. The weighted distance between the model and the client model iteratively updates the parameters between the server model and the client model while focusing on the distance. Chuan Chen et al. [34] proposed a general federated graph learning framework, i.e., Federated Graph Learning (FedGL), which can collaboratively store graph data in different clients to train high-quality graph models while protecting data privacy, coping with heterogeneity between clients. Considering the complementarity of graph data, we propose to discover and exploit global self-supervised information. The process of global self-monitoring discovery and use enables each customer's information to flow and be shared in a privacy-preserving manner, thereby mitigating heterogeneity and exploiting complementarity. Finally, extensive experimental results on node classification tasks show that FedGL significantly outperforms centralized methods, simple joint methods, and local methods, fully validating the effectiveness of FedGL. Han Xie et al. [35–37] proposed a Graph Clustering Federated Learning (GCFL) framework to dynamically find clusters of local systems according to the gradients of GNNs, and theoretically demonstrate that such clusters can reduce the disparity between the structure and features of graphs owned by local systems. Qualitatively, the techniques developed using GCFL allow multiple data owners to hold structures and feature non-IID graphs to collaboratively train powerful graph classification neural networks without direct data sharing.

2. Preliminary

Our work is based on graph convolutional neural networks and federated average algorithms (FedAvg). This section begins with a brief overview of FedAvgs, including the local model update, the communication strategy, and global model aggregation, and then introduces classical graph convolutional neural networks.

2.1. Basic Federated Learning

As a pioneering work in federated learning, the primary training method of FedAvg has become the default method of federated learning. In each round of communication, the client downloads the initial model parameters from the global model and trains the local model through their respective local data. Parameters are then sent back to the terminal server. The local model parameters uploaded are aggregated on the terminal server to obtain a new global model. Only the model parameters are transmitted during the communication between the local model and the global model. Federated Average (FedAvg), the most classic algorithm in federated learning algorithms, is the most widely used algorithm at this stage. The algorithm in this article is also based on the improvement

of this algorithm. The basic idea of FedAvg comes from a distributed learning system composed of a parameter server and multiple local clients. Specifically, suppose there is a federated system model, which includes a parameter server and K clients participating in federated learning. At the beginning of the Tth communication, the parameter server distributes the current global model parameters ω_t to local models ω_t^k of each federated learning client, and each local model $k \in K$ then calculates the gradient once, $g_t^{k(1)} = \nabla_{\omega_t^k} \sum_{(x_i,y_i)\in D_k} \ell(\omega_t^k, (x_i, y_i))$, where D_k is the dataset that has $n_k = |D_k|$ data on the client k, ell is the loss function, and $\nabla_{\omega_t^k}$ represents the gradient symbol of ω_t^k. The terminal server collects all gradients and applies the weighted average method to update the parameters. In this updating mode, each participant is treated equally. However, not every local participant's performance is the same. There are pros and cons, and the average will damage the performance of the global model. The errors come from many aspects, such as random errors caused by model instability, systematic errors caused by machine performance, flawed models, or noise in the data. Although the average value of countless measurements will inevitably tend to approach the global solution, it is impossible to make numerous measurements in practice. The average update aggregation method cannot assign reasonable weights to models with different errors. At this time, it is necessary to process each client differently, taking fully into account the noise of datasets participating in training in different clients.

2.2. Graph Convolutional Networks

Artificial intelligence algorithms have been completely "embedded" in daily life, handling various tasks, image classification, video processing, speech recognition, and natural language understanding. The data in these tasks are usually in Euclidean space. However, real life also contains a large amount of data in non-Euclidean space, that is, data with irregular structures, and these data contain complex and interdependent graph node relationships, which conventional artificial intelligence algorithms cannot handle effectively. Therefore, for non-Euclidean spatial data, graph convolutional neural networks have become another hot research direction in artificial intelligence. They are widely used in life. Many graph convolutional neural networks for processing graph data have appeared in recent years, and they are widely used in various fields, such as chemical molecules, physics, social sciences, knowledge graphs, recommendation systems, and neuroscience. Similar to Convolutional Neural Networks (CNNs) and Multilayer Perceptrons (MLPs), Graph Neural Networks (GCNs) are also trained continuously. Each node on different layers can learn new feature representations, and these new features then pass classifiers to yield output results.

3. Proposed Method

This section first introduces the proposed graph convolutional neural network with a cross-layer fusion structure as the local federated learning model. A federated aggregation algorithm based on an attention mechanism is then proposed to generate an efficient global model. Figure 2 is the federated learning system framework proposed in this paper.

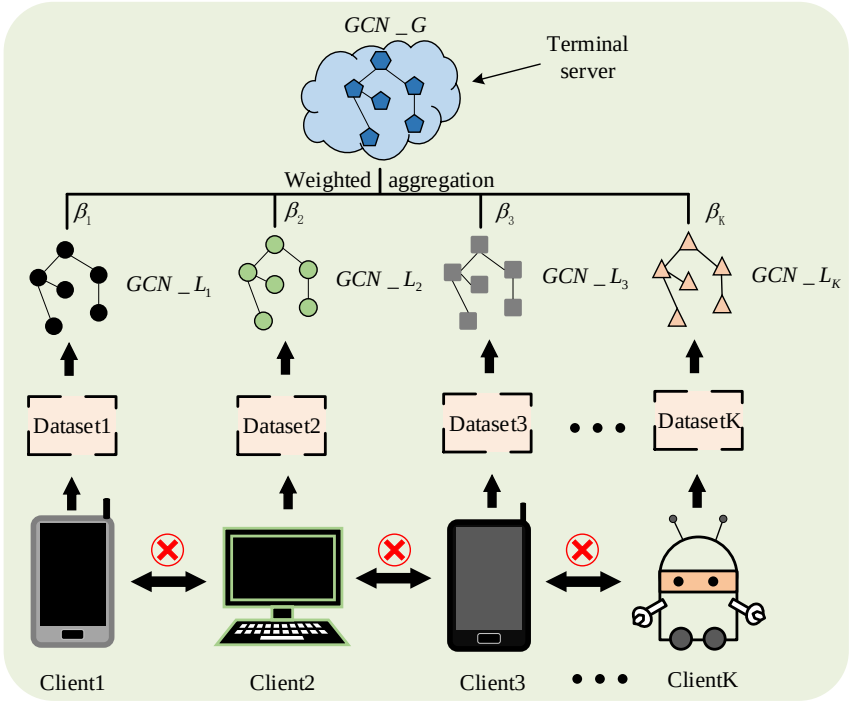


Figure 2. The federated learning framework proposed in this paper.

3.1. The Graph Convolutional Neural Network Architecture

To give the FedGCN the ability to process non-Euclidean spatial data, this section proposes a graph convolutional neural network as a local model of the federated learning client. The GCN is mainly divided into three parts: a convolutional graph layer, a graph pooling layer, and a fully connected layer. The graph convolution layer aggregates the features of the node and its neighbor nodes through the graph convolution operation. The graph convolution layer extracts the effective features of the non-Euclidean spatial data; the graph pooling layer executes the graph pooling operation and selects the best values of various features after the graph convolution layer as representatives to represent new node features. The graph convolution layer output's feature dimension is effectively reduced, as subsequent graph convolution operations or fully connected layer classification provide high-quality parameters. In the fully connected layer, after several graph convolution and graph pooling operations, more efficient features are obtained. Unlike the classic graph convolutional neural network, which only relies on the SoftMax function to output classification results, three fully connected layers are used here to complete the task of graph classification. The specific process is shown in Figure 3.

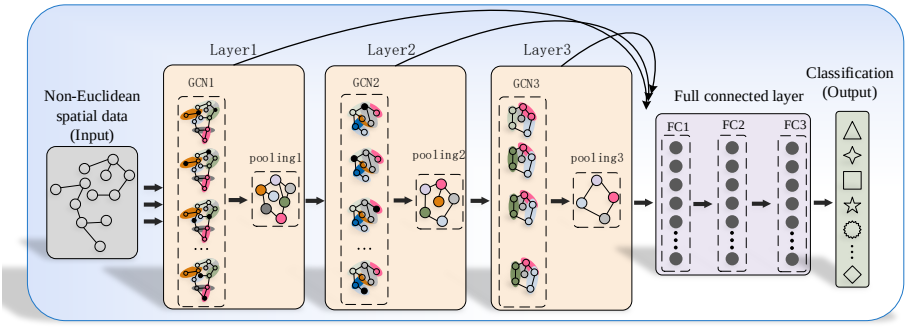


Figure 3. The graph convolutional neural network proposed in this paper.

3.2. The Graph Convolutional Neural Network Architecture

Assuming that there are k clients in total in federated learning, the local undirected graph structure data owned by the i_{th} clients is $G_k(V, E, A)i \in k$. The set of nodes in the graph structure G_i is $v_i \in V$, the edge set between nodes is $e_i, j = (v_i, v_j) \in E$, A is a symmetric matrix containing only 0 or 1, which represents the adjacency matrix of the graph and defines the interconnection relationship between nodes, and the feature on the v_i node is $x_i \in X$. The graph convolutional layer is defined first. Equation (1) is used to aggregate neighbor node information and extract highly generalized effective node features.

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

where W^l is the layer l weight parameter that can be trained, and $\tilde{A} = A + I$ represents the addition of the identity matrix to the original adjacency matrix to contain its node information. H^0 represents the initial input data characteristics, and $\tilde{D}$ is the degree matrix, where $\tilde{D}_{(i,i)} = \sum_j \tilde{A}_{(i,j)}$.

3.2.1. Graph Pooling Layer

A large number of effective features are extracted from the above graph convolutional layer. However, the aggregation collects adjacent node features, and the features may be similar or repeated. It leads to information redundancy and increases computing costs. In order to obtain highly generalized node features, a typical pooling operation in ordinary convolutional neural networks is required. Therefore, we add the graph pool layers to the GCN. The graph pooling layer adaptively selects the node features after the graph convolution operation to form a new but smaller graph.

First, all node features are projected into one-dimensional data through a trainable variable P. Moreover, the TopK pooling operation is performed according to the generated one-dimensional data to select the node with the highest score. Equation (2) shows the pooling operation.

$$\begin{cases} y_i = \frac{x_i^l p^l}{\|p^l\|} \\ i_n = top_n(y_i, n) \\ x_i^{l*} = (x_i^l \odot tanh(y_i))_{i_n} \\ A^* = A_{i\_n, i\_n} \end{cases} \tag{2}$$

where $\| \cdot \|$ represents the 2 norms, y_i represents the one-dimensional vector output of x_i^l after the trainable parameter ρ, $top_n(\cdot)$ selects the index i_n of the highest score from the given input vector, and $\odot$ represents the element-wise multiplication of the corresponding position of the vector, assigning the corresponding weight $tanh(y_i)$ to x_i^l. This paper also use a simple multi-layer perceptron to train. The tanh function enables the trainable parameter p to be trained by back propagation. In the absence of tanh, the projection vector p will produce a discrete output and cannot be trained by back propagation.

3.2.2. Fully Connected Layers

After the previous three groups of graph convolution and graph pooling operations, effective and robust summary features are obtained. However, the feature matrix is not convenient for predicting classification tasks, so a classifier is needed to classify or predict these features to output the final result. As a result, this paper sets three fully connected layers at the end of the network structure. Before inputting the extracted features into the fully connected layer, some preprocessing needs to be performed on the features, that is, cross-layer fusion and dimensional changes. (1) As the graph pooling operation will reduce the number of nodes, some useful information will inevitably be lost, so the cross-layer aggregation of features is necessary. This operation can extract the feature of different processing scales (different graph convolutional layers and graph pooling layers).

In addition, for graphs with a small number of nodes, the information can be effectively preserved. Otherwise, these small graphs' nodes might be quickly discarded. (2) To input high-dimensional feature data into the fully connected layer, this paper flattened the features. This operation saves the final graph node features at a fixed size (the same as the number of neurons in the fully connected layer). The details are shown in Equation (3).

$$\begin{cases} v^l = (\frac{1}{N^l}\Sigma_{i=1}^{N^l} z_i^l) \| (MAX(z_i^l)_{i=1}^{N^l}) \\ V_{fc} = \Sigma_{l=1}^{L} v^l \end{cases} \tag{3}$$

A traditional CNN will perform a single average pooling or a maximum pooling operation before inputting the features extracted by convolutions into the fully connected layer. Unlike a traditional CNN, our method uses Equation (3) to splice two pooling results together. That is, $\|$ represents the concatenate operation. First, average pooling and maximum pooling are performed on the node to obtain feature z_i^l after the graph convolution and graph pooling operations of each layer, respectively, and the two results are concatenated. After these steps are completed, the results obtained from each layer are summed to achieve the effect of cross-layer fusion. In Equation (3), N^l represents the number of nodes, $MAX(\cdot)$ represents the maximum pooling operation, and V_{fc} is the feature finally input to the fully connected layer.

3.3. The Training of GCN

The graphic convolutional neural network training is composed of three parts: a non-linear activation function, a loss function, and an optimizer. The following is a detailed introduction.

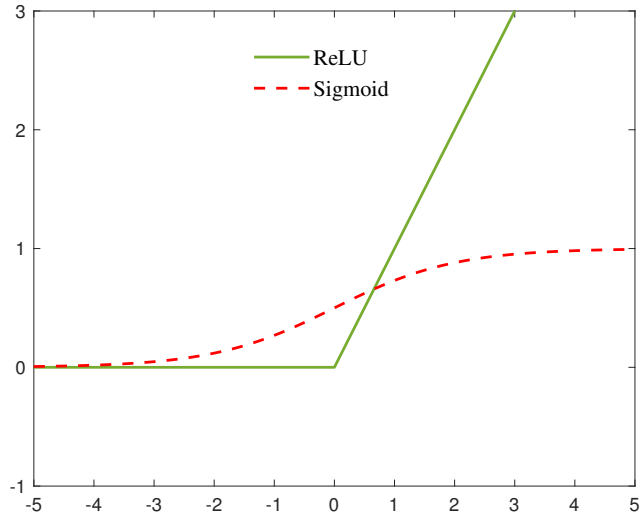
3.3.1. The Activation Function

The input operations of each layer of the graph convolutional neural network, namely, the node features and the corresponding weight matrix, are linear. This paper applies the GCN to the graph structure classification task. This type of classification task is non-linear. Therefore, the non-linear activation function is the critical factor that determines whether the GCN in this paper is effective. This paper uses the ReLU activation function and sigmoid function in different parts of the network. The ReLU activation function acts on the graph convolutional layer, the graph pooling layer, and the first two fully connected layers. The sigmoid function is used in the final fully connected layer to output the final classification result, as shown in Figure 4.

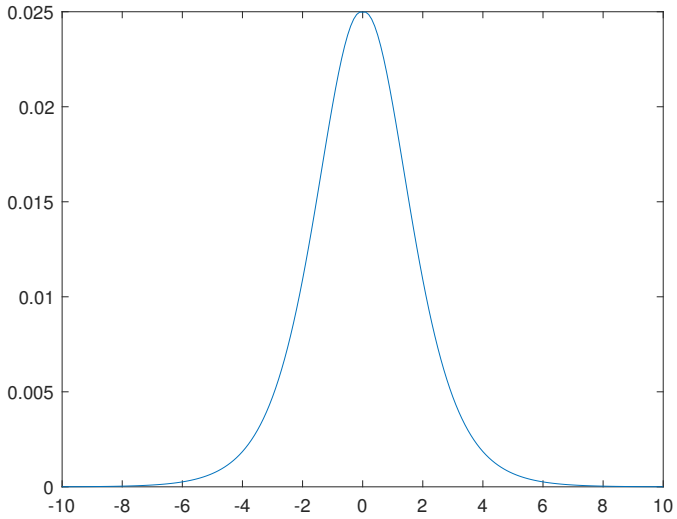
$$sigmoid(x) = \frac{1}{1+e^{-x}} \tag{4}$$

$$ReLU(x) = max(0, x) \tag{5}$$

From Figure 4 and Equation (4), it is not difficult to see that the output value range of the sigmoid function is between 0 and 1. It has good symmetry and is convenient for derivation. It can output smoother values for classification tasks. However, the sigmoid function's partial derivative will disappear when the input value is very large or very small, so the sigmoid is only used in the last fully connected layer to output the classification results. The ReLU activation function is used in the middle layer of the graph convolutional neural network to prevent the gradient from disappearing.



(**a**) ReLu function



(**b**) Sigmoid function

Figure 4. Schematic diagram of the activation function.

3.3.2. The Loss Function

The loss function is an important indicator for guiding and evaluating model training. A good loss function can accurately quantify the deviation between the estimated value and the actual label. Assuming that the data of the kth client contains the feature x and the label y, namely, $D_k(x, y)$; the labels have C categories, namely, $y = [y_1, y_2, ...y_c]$; the labels are coded by one-hot representation; and only one digit is used for effective representation. In this paper, the GCN model is denoted as G, and the node feature X is input into the model to obtain the output $G(X)$. The cross-entropy loss function used in this paper is shown in Equation (5).

$$l_k(y_i^k, G(x_i^k)) = -\Sigma_{i=1}^{n_k}[y_i^k log G(x_i^k) + (1 - y_i^k) log(1 - G(x_i^k))] \quad (6)$$

where n_k represents the amount of data owned by the k_{th} client, and l_k represents the corresponding loss function.

Compared with a simple loss function such as the mean squared error, the cross-entropy loss function can better adjust weights. When the absolute error (the deviation between the predicted value and the true value, that is, $y - y^{'}$) is very large, the model converges very slowly or does not converge at all. Figure 5 shows the relationship of the absolute error and gradient between the cross-entropy loss function and the Mean Squared Error (MSE) loss function during the training process. Suppose there is a set of data $(x, y) : y = sigmoid(\theta \times x + b)$, where $x = 1, y = 1$, θ is constantly changing, and b is always fixed at 0.2. When ω changes, the absolute error and the gradient of ω are recorded. The absolute error abs_error is the abscissa, and the gradient is the ordinate. It can be clearly seen in Figure 5 that the absolute error in the cross-entropy loss function is proportional to the gradient. That is, the smaller the absolute error, the smaller the gradient. As the absolute error in the mean square error loss function increases, the gradient is distorted, which will cause the model to fail to converge. Based on these advantages, this paper uses the cross-entropy loss function.

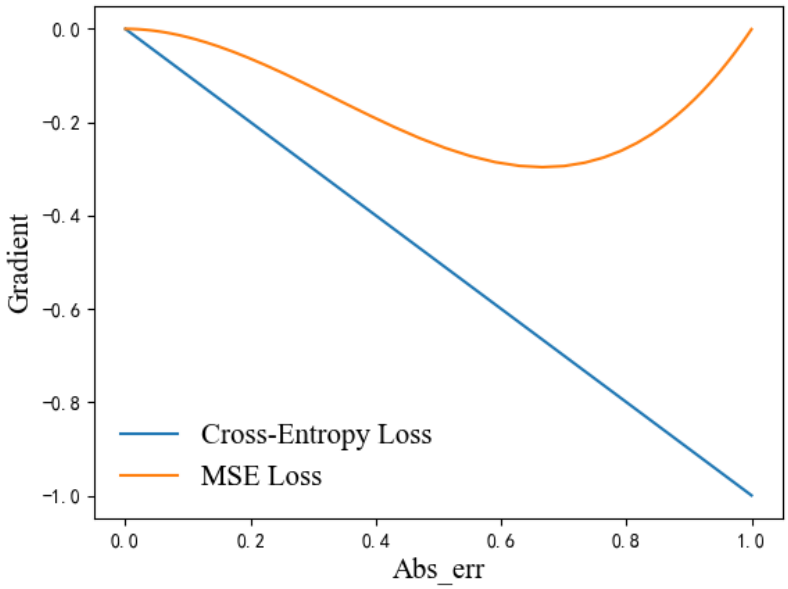


Figure 5. The relationship between the training gradient and the absolute error.

3.3.3. Optimizer

The previous section defines the loss function to continuously evaluate the current model's pros and cons during the training process. That is, the smaller the loss function, the better the model is in general. However, if there is no tool to find the optimal solution of model parameters, no matter how good the loss function is, it cannot play its role. The optimizer adjusts model parameters to minimize the loss function. Stochastic Gradient Descent (SGD) is currently the most commonly used optimizer for neural networks or other machine learning algorithms; it is shown in Equation (7):

$$\tilde{\theta} = \theta - \eta \cdot \nabla l_k(y^k, G(\theta, x^k)) \tag{7}$$

where θ represents the parameter to be optimized, $\tilde{\theta}$ represents the updated θ parameter, $l_k(\cdot)$ represents the loss function, ∇ represents the gradient, and η represents the learning rate or step size, which limits the speed of network learning. The SGD algorithm only calculates the gradient of a small part of the sample in each iteration, so the learning speed is faster each time. SGD only uses a small sample to represent all samples to update θ, so it is easy to converge to the local optimum. Adam [38] is an optimization method that can adaptively adjust the learning rate to adapt to various parameters, effectively preventing the training parameters from being trapped in the local optimum. It is shown in Equations (8) and (9) is the parameter update equation.

$$\begin{cases} m_t = \gamma_1 m_{t-1} + (1-\gamma_1) g_t \\ v_t = \gamma_2 v_{t-1} + (1-\gamma_2) g_t^2 \end{cases} \tag{8}$$

$$\begin{cases} \tilde{m}_t = \frac{m_t}{1-\gamma_1^t} \\ \tilde{v}_t = \frac{v_t}{1-\gamma_2^t} \\ \tilde{\theta} = \theta - \eta \cdot \frac{\tilde{m}_t}{\sqrt{\tilde{v}_t}+\epsilon} \end{cases} \tag{9}$$

where m is the first-order moment estimation of the gradient, that is, the mean value of the gradient. v is the second-order moment estimation of the gradient, namely, the biased variance of the gradient. g is the gradient, and t represents the number of iterations of current learning. $g_t^2 = g_t \odot g_t$, and $\odot$ represents the element-wise multiplication of the corresponding position of vectors. $\gamma_1, \gamma_2 \in [0, 1)$ is a set of hyperparameters, according to the actual experience provided in the paper [31]. Here, $\gamma_1 = 0.9$ and $\gamma_2 = 0.99$. $\tilde{m}_t$ and $\tilde{v}_t$ are the mean and biased variance of the corrected gradient. As the moment estimation of the gradient does not require additional memory, it will not increase the pressure on the memory. Moreover, after Adam has been biased and corrected, each iterative learning rate has a certain range, making the parameters relatively stable.

Figure 6 shows the changes of four mainstream optimizer loss functions when the second-generation GCN model is trained with 1000 epochs on the Cora dataset. It can be clearly seen that the Adam optimizer has the best effect in reducing loss. When SGD, with or without momentum, is applied to graph data and graph neural network models, they both fail. Because of Adam's advantages mentioned above, this article uses Adam as the network model's optimizer.

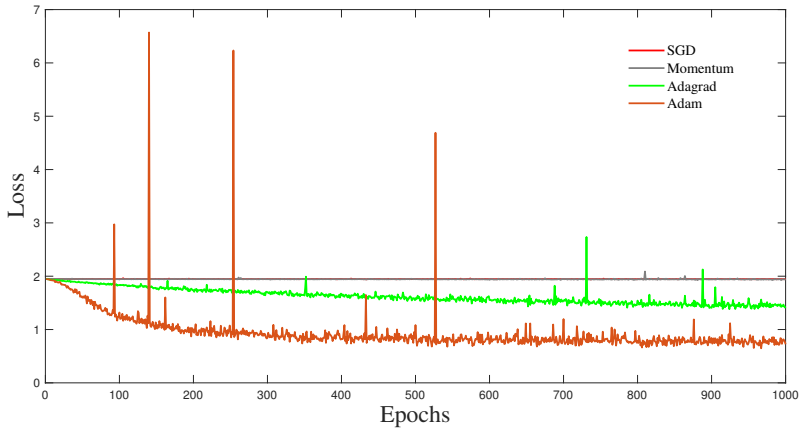


Figure 6. The performance of the Adam optimizer in the loss function.

3.4. The Federated Aggregation Based on Attention Mechanisms

Sections 3.1 and 3.2 introduce the GCN model structure and its training method used by a single client of the federated learning system in this paper. The GCN gives FL the ability to process non-Euclidean spatial data. The federated learning system can jointly train multiple graph convolutional neural networks without sharing data. Because of this characteristic of FL, it can fully extract the training experience of each local GCN model in the training process, and finally train an efficient global model. However, the traditional federated average algorithm, similar to FedAvg, cannot evaluate the merits and defects of each local model well but makes unified average processing, which is rough and may lead to unsatisfactory training results of the global model. Therefore, this chapter proposes an attention mechanism to develop attention weights that can be adjusted online for each local model. By aggregating each local model according to this weight, the influence of potential noise can be reduced. These noises may come from the data of each client, or from the model itself, and so on. The contents of this chapter are as follows.

The federated learning system based on the graph convolutional neural network proposed in this chapter includes K federated learning participants and a terminal server. The federated learning client processes its own graph data by training a local graph convolutional neural network (defined in Section 3.1). Afterwards, the relevant parameters are merged into the terminal server, and after continuous iterative training, a network model with excellent performance is finally obtained. Federated learning is mainly divided into two parts: (1) client local model training; (2) terminal server fusing local model parameters. The model training and parameter uploading and downloading modes of FedGCN are similar to those of FedAvg. The algorithm pseudocode of FedGCN is shown in Algorithm 1.

Algorithm 1 FedGCN.

Input: B is the minimum batch size of the local model (min batch size). E is the number of iterations of the local model. α is the learning rate. m is the number of clients participating in federated learning. p_k is the dataset index of the kth client.
Output: Global weights $\theta^G_{l,t+1}$
Global model optimization://the number K of participants in FL, datasets D_k,the fraction of clients C
Initialize the global model parameter θ^{global}
for each round $t \rightarrow 1, 2, \ldots$ **do**
  S_t = (random set of $\max(C \cdot K, 1)$ clients)
  for each client $k \epsilon S_t$ **in parallel do**
  receive the local loss θ^k_{t+1} and operate the attention mechanism $att(.)$(Equation (11))
    calculate the confidence β^l_k (Equation (12))
    Information fusion: update global weights $\theta^G_{l,t+1} = \sum_{K=1}^{K} \theta^{l,t}_k$ (Equation (13))
    Information distribution: pass $\theta^{l,t}_k$ back to the local model
  end for
end for
Local participant updates://local model parameters θ^{local}_k,datasets D_k
  batches = (data D_k split into batches of size B)
  Download the global model optimization parameter θ^t_g to initialize θ^{local}_k
for each local epoch i from $1 \rightarrow E$ **do**
  for batch b in batches **do**
    $\theta = \theta - \alpha \nabla \ell(\theta; b)$
  return $\theta, \nabla \ell(\theta; b)$ to step3
end for

The following parts focus on the aggregation mode of each partial model in the terminal server. The most important part of federated learning is the federated optimization of the terminal server, which aggregates client models during the federated optimization process. The existing federated learning adopts the average aggregation method without considering each local model's characteristics in order to further increase the accuracy of federated learning on non-Euclidean spatial data. This section proposes a federated aggregation method based on the attention mechanism, which can be adjusted online. The global model uses optimal parameters learned from each local model's individual training experience to generate an optimal global model. The algorithm pseudocode of FedGCN is shown in Algorithm 1.

The parameters uploaded by the local model to the terminal server are only weights of the training parameters in the graph convolutional neural network. Before federated learning, the clients are randomly selected, shown in Equation (10).

$$n\_fed = max(C \cdot K, 1) \tag{10}$$

where n_fed is the number of clients in federated learning. There are K clients in federated learning. The proportion of clients in each round of calculation is C. If the amount of data on the client is known, the clients with a larger amount of data are preferred to participate

in federated learning. It is a simple reward mechanism. When starting training, clients initialize the local model parameters θ_k^l. The local model parameters are then uploaded to the terminal server to initialize the global model. (The global model GCN_G has the same structure as the graph convolutional neural network built in Section 3.1.) After the global model and the local model have been initialized, in order to aggregate an excellent global model on the terminal server, each client needs to be evaluated to obtain model confidence $\beta_k = \{\beta_k^0, \beta_k^1, \ldots, \beta_k^l\}$ (β_k is initialized to 1/K) in each layer of the GCN constructed in this article. That is, the attention mechanism is performed on the kth local graph convolutional neural network model GCN_L_k. After this operation, the parameter confidence of each layer of GCN_L_k is output through a SoftMax function, which is used to aggregate the global model and other local models.

To dig deeper into the relationship between the global model GCN_G and each local model, this paper uses an attention mechanism with a trainable parameter p_k^l. The attention mechanism is a single-layer perceptron neural network, the importance of the local GCN_L to the global GCN_G is quantified in mathematical form, and at the same time, the federated aggregation can adjust the aggregation strategy online according to p_k^l, as shown in Equation (11).

$$a_*^l = att(\theta_k^l, \theta^l) = p_k^l[\theta_k^l \| \theta^l] \tag{11}$$

where $att(\cdot)$ represents the calculation function of the attention mechanism, θ_k^l represents the trainable parameter of the kth local GCN_L_k of the lth layer, and θ^l represents the trainable parameter of the global GCN_G of the lth layer. The symbol $\|$ represents the matrix concatenate operation. The attention coefficient obtained by the function $att(\cdot)$ may have a large difference in value. In order to make the coefficients easy to be used in the parameters of each layer without affecting the convergence speed because of a possible parameter transform in order of magnitude, the SoftMax function is used to standardize the attention coefficient, as shown in Equation (12).

$$\beta_k^l = SoftMax(att(\theta_k^l, \theta^l)) = \frac{exp(a_k^l)}{\Sigma_{k \in K} exp(a_k^l)} \tag{12}$$

where β_k^l represents the parameter confidence of the lth layer of the k_{th} local graph convolutional network, and the SoftMax function can ensure that the confidence sum is 1, that is, $\Sigma_{k=1}^K \beta_k^l = 1$. After obtaining the parameter confidence of each local model by Equation (12), the terminal server needs to update the global GCN_G model parameters according to these confidences and local GCN_L_k model parameters, as shown in Equation (13).

$$\theta_G^{l,t+1} = \Sigma_{k=1}^K \beta_k^{l,t} \theta_k^{l,t} \tag{13}$$

where $\beta_k^{l,t}$ represents the attention weight coefficient assigned to the kth participant model at time t, and $\theta_G^{l,t+1}$ represents the first layer parameter of the global model aggregated at time $t+1$. $\theta_k^{l,t}$ represents the parameters of the local model.

4. Experiment and Discussion

This section performs graph classification tasks on nine public graph structure datasets, *D&D*, ENZYMES, IMDB-BINARY, REDDIT, PROTEINS, GITHUB-STARGAZERS, COLLAB, IMDB-MULTI, NCI1, to verify the method proposed in this chapter and compare it with conventional graph convolutional neural networks. The most remarkable common datasets have a regular spatial structure, such as MINST and CIFAR, which can be represented by a matrix, and a traditional convolution neural network is handled more efficiently. However, many data in life do not have regular spatial structures, such as recommendation systems, molecular structures, and so on. The connections of each node in these maps are different and have irregular structures. For these irregular data objects, the graph convolution is better than the traditional convolution neural network in dealing

with these irregular structure data. Therefore, we use six public datasets, all of which are datasets with irregular spatial structures, to prove the accuracy and versatility of the federated learning framework proposed in this paper in dealing with non-Euclidean spatial data classification.

It is worth mentioning that clients in federated learning need to conduct synchronous training and update parameters on different clients in principle. In order to simulate this federated learning setting, the experimental part of this paper adopts serial training for each client and uploads its parameters to the terminal server after each training is completed. In the training process, the number of training iterations $local\_ep$ of the local model is set as 10, and the batch size $local\_bs$ of the local model's training data is set as 60. The learning rate lr defaults to 0.005. In this article, the same local model is set for each local client, but different attention weights are assigned to aggregate them.

4.1. The Dataset Introduction

[1] *D&D* extracts 1178 high-resolution proteins in a subset of a protein database. Each graph represents a protein. The nodes in the graph are amino acids. If the distance between the two nodes is less than 6 angstroms (1 angstrom equals 0.1 Nano), then one edge is used to connect them. The task is to classify proteins as enzymes or non-enzymes.

[2] ENZYMES is a protein tertiary structure dataset composed of 600 enzymes from the BRENDA enzyme database. The task is to classify each enzyme into six categories correctly. This category is classified according to the type of enzyme reaction.

[3] PROTEINS is also a protein dataset. The nodes represent the secondary structure elements of proteins, and the edges represent the biological significance between different types of proteins. The task is to determine whether a protein is an enzyme or not.

[4] COLLAB is a subset of a scientific collaboration dataset. A researcher corresponds to a node, and an edge corresponds to the cooperative relationship between researchers. The task is to determine whether a researcher studies high-energy physics, condensed matter physics, or astrophysics.

[5] REDDIT-BINARY is a balanced dataset, where each graph corresponds to an online comment post, and the node corresponds to the user. If one node responds to another node's comment, there is an edge between the two nodes. The task is to determine whether the graph belongs to a question–answer forum or a discussion-based forum.

[6] GITHUB-STARGAZERS contains a diagram representing the relationship network of GitHub users. These users are divided into the popular machine learning knowledge base, and the other is the web development knowledge base.

[7] IMDB-BINARY is a movie collaboration dataset where we collected actor/actress and genre information of different movies on IMDB.

[8] IMDB-MULTI is multi-class version of IMDB-BINARY and contains a balanced set of ego-networks derived from Comedy, Romance, and Sci-Fi genres.

[9] NCI1 represents one balanced subsets of data sets of chemical compounds screened for activity against non-small cell lung cancer.

Before applying these data to federated learning training, the dataset is divided into N sub-datasets according to the number of federated learning clients N. $\frac{1}{10}$ of graphs is used as the test set. The rest is the training set. If graphs are not divisible by 10, it is rounded up. The experimental part of this paper divides the dataset into three clients, according to McMahan et al. [1]. Each client receives $N_G \times \frac{9}{10} \times \frac{1}{3}$ samples. Datasets Statistics summarize from the Field, Graphs, Classes, Average Nodes and Average Edges to Table 1:

Table 1. Datasets Statistics.

Datasets	Field	Graphs	Classes	Average Nodes	Average Edges
D&D	Bioinformatics	1178	2	284.32	715.66
ENZYMES	Bioinformatics	600	66	32.63	62.14
PROTEINS	Bioinformatics	1113	2	39.06	72.82
COLLAB	Social networks	5000	3	74.49	2457.78
REDDIT-BINARY	Social networks	2000	2	429.63	497.75
GITHUB-STARGAZERS	Social networks	12,725	2	113.79	234.64
IMDB-BINARY	Social networks	1000	2	19.77	96.53
IMDB-MULTI	Social networks	1500	3	13	65.94
NCI1	Small molecules	4110	2	29.87	32.30

4.2. Evaluation Method

This paper uses accuracy to evaluate FedGCN. True Positive (TP) represents a positive sample that is correctly predicted by the model. False Positive (FP) represents a positive sample that is predicted to be negative by the model. False Negative (FN) represents a negative sample that is predicted to be positive by the model. True Negative (TN) represents a negative sample that is predicted to be negative by the model. The equations for accuracy and recall are shown in Equation (14) below:

$$Accuracy = \frac{TP}{TP + FP} \tag{14}$$

The accuracy rates mentioned in this article are all calculated by this formula. The quality of the evaluation index lies in whether it can intuitively reflect the performance of the algorithm. It is often necessary to design different evaluation indicators in different tasks, and sometimes multiple indicators are needed to reflect the objective situation in collaboration.

4.3. Validation Experiment of FedGCN Attention Mechanism

In this section, the FedGCN proposed in this paper and the classical FedAvg algorithm are used to carry out comparative experiments on six datasets. The local models of the FedAvg and the FedGCN are the graph convolutional neural networks built in Section 3.1. The hyperparameters and the training method of FedGCN are consistent with FedAvg, to verify the effectiveness of the aggregation algorithm with the attention mechanism in the FedGCN.

As the FedGCN evaluates each of the local model's parameters through the attention mechanism, the corresponding confidence coefficient is obtained to aggregate the parameters to generate the global model. The global model can then extract the training experience of optimal local models in this way. As shown in Figure 7, the accuracy of the FedGCN on different datasets is higher than that of FedAvg. It verifies the effectiveness of the FedGCN. Compared with the FedAvg algorithm, the attention-based parameter aggregation method does an excellent job of evaluating each local model to be uploaded to the terminal server and formulating the corresponding confidence subsequently. Finally, the new global model is aggregated by the dot product of the local model parameters' confidence. This method can well reduce the influence of inferior local model parameters and improve the fault tolerance and accuracy of FedGCN.

4.4. FedGCN Accuracy Verification Experiment

In order to verify that the algorithm FedGCN proposed in this paper has excellent accuracy, experiments were carried out on various non-Euclidean spatial datasets. Figure 7 shows the performance comparison of FedGCN, GCN, and GIN models. The FedGCN local model is constructed in Section 3.1, and the GCN also corresponds to the graph convolutional neural network constructed in Section 3.1. Experiments verified the effectiveness of the FedGCN in processing non-Euclidean spatial data. Moreover, compared with the classical GCN model, the GCN constructed in this paper, by adding a TopK pooling layer,

a full connection layer, and a cross-layer fusion module, has better accuracy and stability. The experimental analysis is mainly based on each algorithm's accuracy performance on various datasets (the accuracy is obtained by testing on the test dataset after each round of communication).

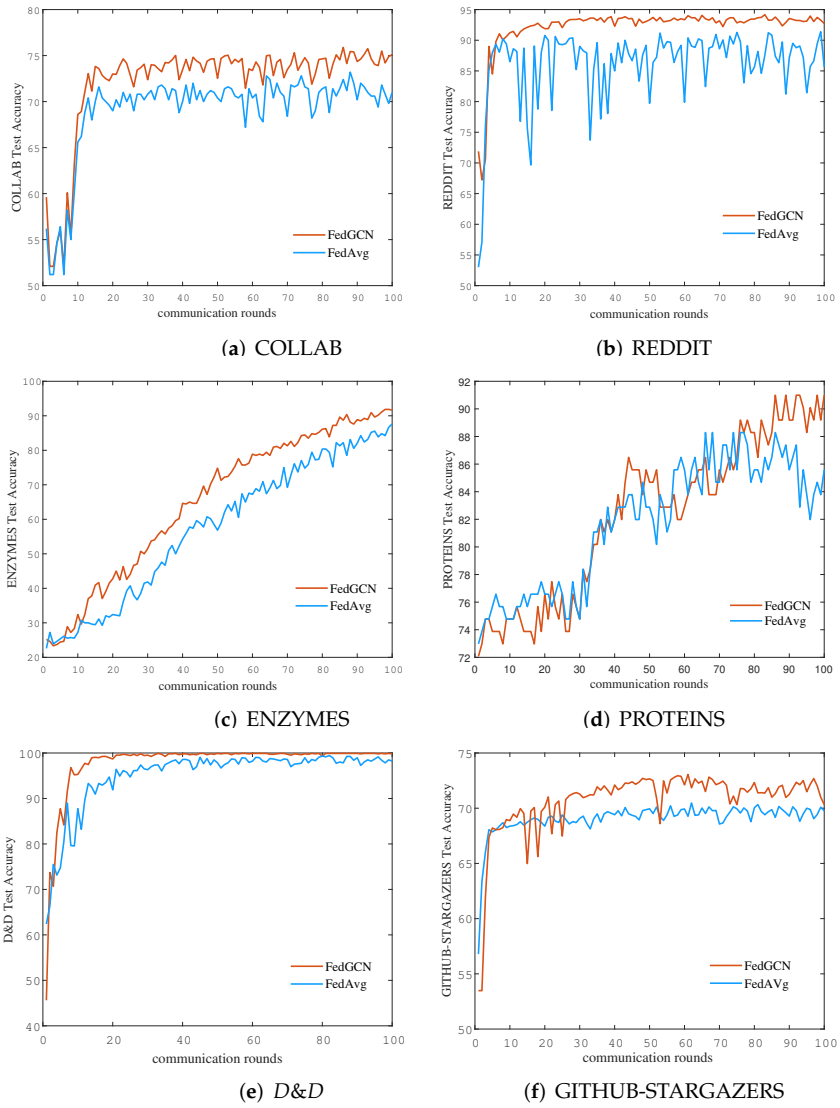
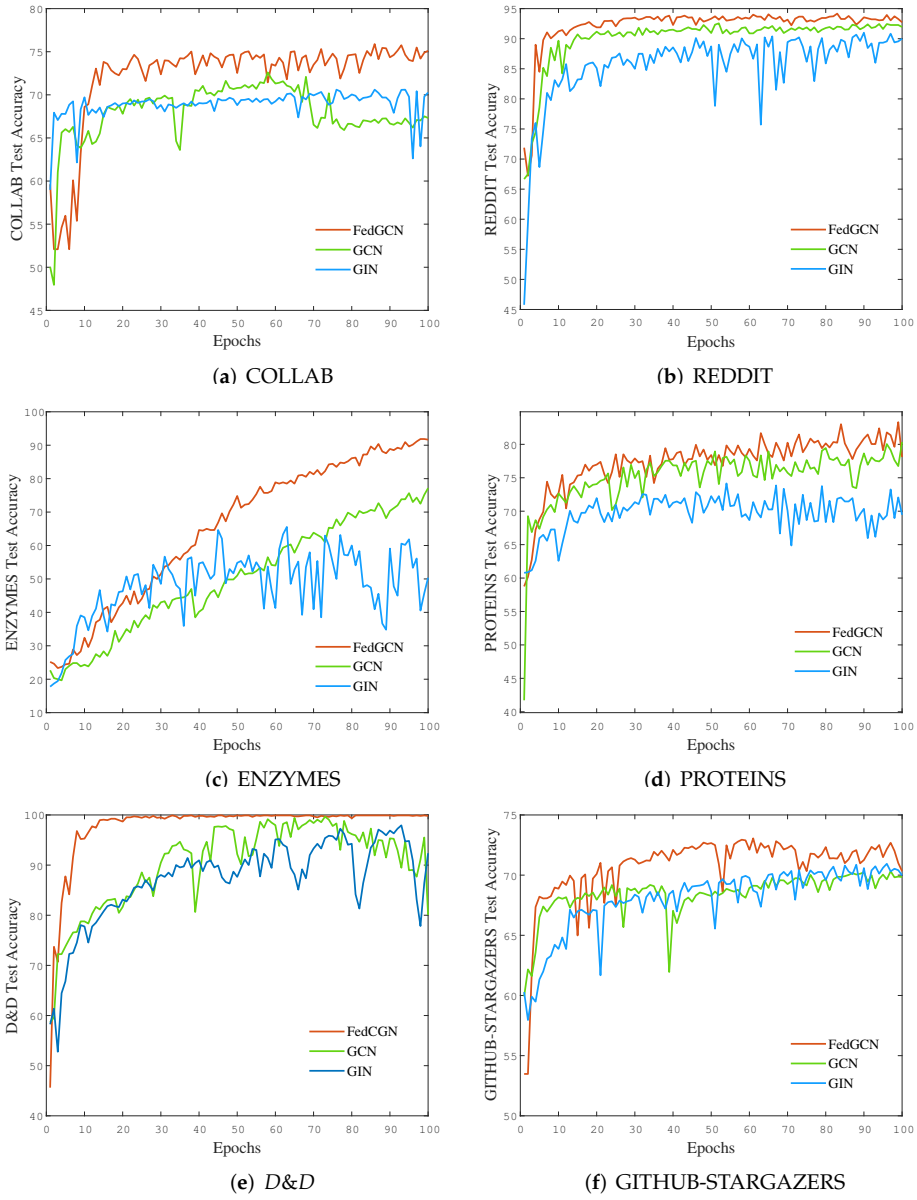


Figure 7. Accuracy experiment of the FedGCN algorithm on six datasets.

Figure 8 shows the accuracy experiments of the three models on the six datasets—COLLAB, REDDIT, ENZYMES, PROTEINS, *D&D*, and GITHUB-STARGAZERS (epochs corresponding to FedGCN are the communications rounds between the local model and the global model). It can be clearly seen that FedGCN has a high accuracy rate regardless of the dataset. It is obvious that FedGCN has high accuracy with respect to any dataset, and the training process is relatively stable compared with the other two GCN models. There is no significant sign of decline in accuracy. The accuracy of the GCN model is also relatively higher compared to the newer GIN algorithm. FedGCN adds a TopK pooling layer, a full connection layer, and cross-layer fusion, which can well extract node features, obtain relatively superior local model parameters, and prepare for global aggregation.



(**a**) COLLAB (**b**) REDDIT

(**c**) ENZYMES (**d**) PROTEINS

(**e**) *D&D* (**f**) GITHUB-STARGAZERS

Figure 8. Accuracy experiment of FedGCN algorithm on six datasets.

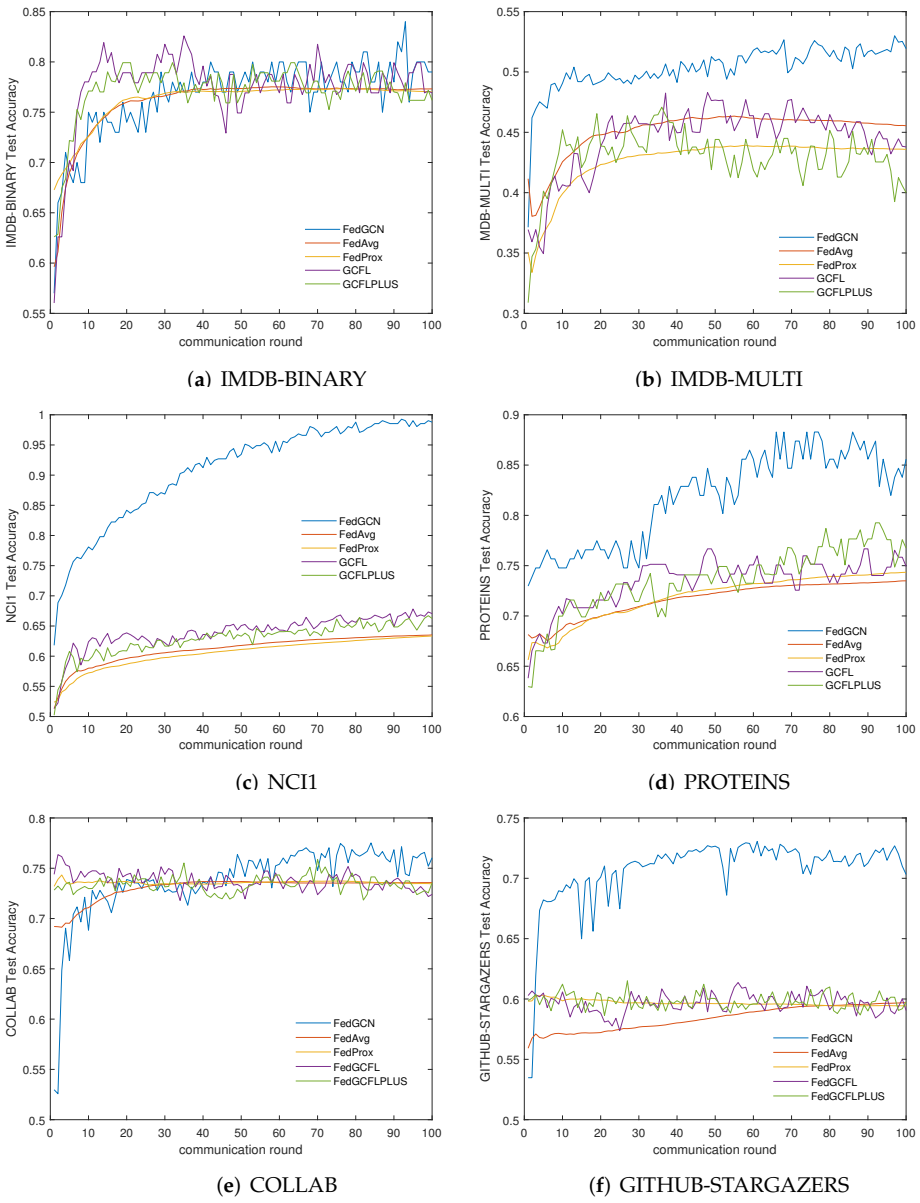
Furthermore, the aggregation algorithm proposed in this paper, which is based on the attention mechanism, can improve the fault tolerance of the FedGCN. The attention mechanism for local model parameters is introduced, thus improving the influence of excellent parameters on the global model and reducing the poor local model's effects. The combination of the GCN and the aggregation algorithm, which is also based on the attention mechanism, improves the FedGCN's ability to process non-Euclidean space data, thus improving the accuracy. We have made a summary comparison, and the specific results are as follows Table 2:

Table 2. Federated learning global model accuracy at different iterations (%).

Model	**FedGCN**		**GCN**	**GIN**
	Communication Rounds			**Epoch**
Dataset	**10**	**20**	**100**	**100**	**100**
COLLAB	68.60	73.00	**75.89**	67.31	70.27
REDDIT	91.22	91.89	**94.17**	91.94	90.00
ENZYMES	32.40	42.78	**91.85**	77.17	50.37
PROTEINS	68.47	75.67	**81.08**	80.34	69.46
D&D	95.28	98.68	**99.90**	80.05	92.37
GITHUB	68.97	69.71	**73.05**	69.84	69.99

The number of training iterations of a conventional GCN epoch is equivalent to the multiplication of the number of local model training epochs $local_{ep}$ and the number of communications with the global model rounds, namely, $Epochs = local_{ep} \times CommunicationsRounds$, where $\equiv$ represents an equivalent symbol. The number of iterations of FedGCN's local model $local_{ep}$ is 10. $CommunicationsRounds$ is 100, and it is not difficult to see in Figure 7 that the accuracy of FedGCN on the graph dataset has reached a relatively high level. In the 10th communication round, the accuracy of the global GCN model in the FedGCN framework is similar to that of the conventional GCN model with intensive training. When the accuracy of the conventional GCN model stabilizes, FedGCN can continue to improve accuracy by increasing communication rounds, which fully demonstrates the superiority of the federated learning method proposed in this paper. Note that the accuracy of the GCN constructed in this paper shows better accuracy in each dataset in most cases.

From the overall view of Figure 9, the FedGCN algorithm proposed in this paper has obvious advantages in dealing with four kinds of data sets. Furthermore, the difference between FedGCN and other algorithms in the four images mainly lies in the different local models used and different aggregation methods, and the accuracy of FedGCN in the four test sets is at a high level, indicating that this paper organically combines TopK pooling layer and cross-layer fusion mechanism, and the GCN model built makes federation learning have higher accuracy and has certain advantages compared with ordinary FedAvg algorithm. This advantage exists even over FedProx. The federated learning framework formed by combining the attention aggregation algorithm with the GCN model built in this paper also has higher accuracy than the new GCFL algorithm. Although GCFLPLUS which improved GCFL based on observation sequences of gradients improves the performance of GCFL, it still can not exceed FedGCN. The graph convolution neural network built in this paper adds TopK graph pooling layer and full connection layer on the basis of the classical model to fully extract node features. Then, in order to prevent the loss of important information, a cross-layer fusion mechanism is proposed, which integrates the features extracted by all the layers before the full connection layer. In order to improve the fault tolerance of federated learning aggregation, and then improve the accuracy of the global model. In this paper, a federated aggregation algorithm that can be adjusted online is proposed. Based on the attention mechanism, this algorithm provides the corresponding attention weight for each client model parameter, and this weight is applied to the parameters of each layer in order to accurately use the training parameters of each local model to get the optimal global model.



(**a**) IMDB-BINARY (**b**) IMDB-MULTI

(**c**) NCI1 (**d**) PROTEINS

(**e**) COLLAB (**f**) GITHUB-STARGAZERS

Figure 9. Comparison of the accuracy of federated learning algorithms.

The advantages of FedGCN can also be clearly seen from Table 3. The table lists the average accuracy of the global model obtained by each algorithm after 100 rounds of communication on different data sets and the highest accuracy in 100 rounds. To a certain extent, the average accuracy can show that FedGCN has a higher accuracy in most communication rounds, and the highest accuracy can represent the best processing performance of FedGCN. These processes benefit from the fact that FedGCN has a good local model, which can not only fully extract the features of the graph structure data, but also fine processing for each client to customize the aggregation weights suitable for each local model.

Table 3. Comparison of the accuracy of federated learning global models (%).

Dataset	Accuracy	Model				
		FedGCN	**FedAvg**	**FedProx**	**GCFL**	**GCFLPLUS**
IMDB-BINARY	Average	**82**	72	72	73	74
	Best	**88**	74	74	77	79
IMDB-MULTI	Average	**77**	76	76	77	77
	Best	**84**	76	77	83	80
NCI1	Average	**50**	45	42	44	42
	Best	**53**	46	44	48	47
PROTEINS	Average	**82**	61	61	61	63
	Best	**88**	63	63	68	67

As shown in Figure 10, the FedGCN was used to test the PROTEINS dataset on different graph convolution models. The graph convolution model includes SAGE [25] GIN [28], MINCUT [33], and the GCN model built in this paper. The blue bars in Figure 9 represent the GCN with the FedGCN settings. The orange bars represent the pure GCN. It can be seen in the figure that the highest accuracy rate of the FedGCN after 100 rounds of communication is higher than that of the GCN model obtained by 100 epochs of iterative training, which shows that the federated learning framework proposed in this paper is effective for graph convolution. The neural network model has a certain versatility and is convenient for embedding in other models, so a variety of graph convolutional neural networks could be combined to process various types of non-Euclidean spatial data. Because FedGCN processes local model parameters more finely through an aggregation algorithm based on an attention mechanism, the aggregation algorithm assigns appropriate weights to each local model parameter, and the fault tolerance of local model aggregation is improved.

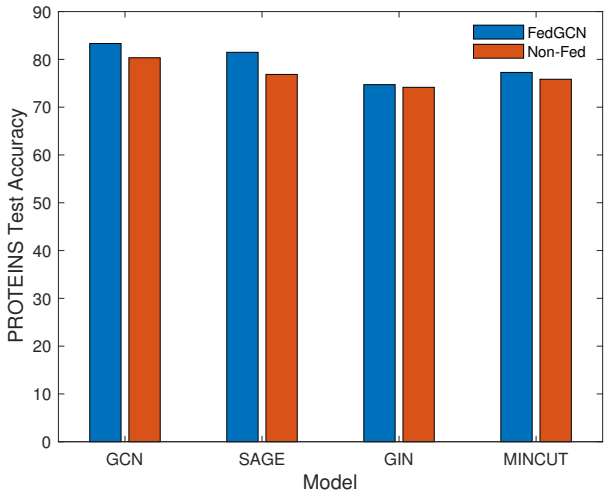


Figure 10. The generality of the FedGCN framework.

5. Conclusions

Federal learning was conceived to provide solutions for isolated data islands, a characteristic of the big data era. However, many problems in this field have arisen. This paper focuses on the following two problems: the accuracy of federated learning algorithms decreases when handling noise data, and federated average algorithms being too rough for the client side. At present, there are few federated learning algorithms that process non-Euclidean spatial data. In view of the fact that the existing federated learning algorithms seldom have the ability to process such data, federated learning based on a graph convolution neural network is proposed in this paper.

In this paper, a graph convolution neural network is constructed as a local model of a client. On the basis of a classical graph convolution neural network, a pool layer and a full connection layer of a TopK graph are added, and this not only improves the feature extraction ability of the local model, but also reduces the amount of calculation required, so the network can output classification results more smoothly. This paper also proposes a cross-layer fusion method to prevent the loss of graph node features. Afterwards, in order to give the federated learning algorithm the ability to process non-Euclidean spatial data, the GCN model constructed in this paper is combined with federated learning. Finally, aimed at the defect that the federated average aggregation algorithm cannot distinguish the parameters of a local model with poor performance, a federated average aggregation algorithm based on an attention mechanism is proposed. The algorithm uses a perceptron neural network as an attention mechanism, which fully considers the importance of each local model parameter participating in federated learning and establishes a corresponding concern coefficient for it, in order to reduce the interference with the model aggregation.

Although the algorithm proposed in this paper can improve problems caused by the average aggregation of the classical federated learning algorithm, there are still issues to be addressed:

[1] Communication cost. The two federated learning frameworks proposed in this paper have improved the local model, leaving a large number of calculations in the client, so as to ensure the effectiveness of upload parameters and finally reduce the number of communications and the communication costs. In the future, model compression can be introduced to further improve the efficiency of this algorithm.

[2] Privacy protection. This paper adopts a privacy protection method that is consistent with the classical FedAvg; that is, the privacy is protected by transmitting the parameters of the model while keeping the original data local. However, this level of privacy protection cannot meet higher-level requirements. More secure privacy protection technologies can be explored to enable federated learning in the future.

Author Contributions: Conceptualization, K.H.; methodology, K.H.; software, J.W., Y.L. and M.L.; validation, J.W., Y.L. and M.L.; formal analysis, J.W., Y.L. and M.L.; investigation, J.W., Y.L. and M.L.; resources, K.H., L.W. and M.X.; data curation, K.H.; writing—original draft preparation, J.W., Y.L., M.X. and L.W.; writing—review and editing, J.W. and Y.L.; visualization, J.W. and Y.L.; supervision, K.H.; project administration, K.H.; funding acquisition, K.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Natural Sciences and Engineering Research Council of Canada, Canada Foundation for Innovation, British Columbia Knowledge Development Fund, and Western Economic Diversification Canada. The research was undertaken, in part, thanks to funding from the Canada Research Chairs program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data and code used to support the findings of this study are available from the corresponding author upon request (001600@nuist.edu.cn). The data are from the open TUDataset (https://chrsmrrs.github.io/datasets/docs/datasets/, accessed on 23 August 2021).

Acknowledgments: Research in this article was supported by the National Natural Science Foundation of China (42075130, 61773219, and 61701244). The key special project of the National Key *R&D* Program (2018YFC1405703) and the financial support of Nanjing Ta Liang Technology Co., Ltd. is deeply appreciated. The authors would like to express heartfelt thanks to the reviewers and editors who submitted valuable revisions to this article.

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclature

Formula symbols and meanings (The order sort appears in this paper)

ω_{t}	Represents the current global model parameters
ω_t^k	Represents the local model of each federated learning client
ell	Represents the loss function
$\nabla_{\omega_t^k}$	Represents the gradient symbol of ω_t^k
l	Represents the number of layers
$\tilde{A} = A + I$	Represents the addition of the identity matrix to the original adjacency matrix to contain its node information
H^0	Represents the initial input data characteristics
$\tilde{D}$	Represents the degree matrix
W^l	Represents the layer l weight parameter that can be trained
$\|\cdot\|$	Represents the 2 norms
y_i	Represents the one-dimensional vector output of x_i^l after the trainable parameter ρ
$top_n(\cdot)$	Represent selects the index i_n of the highest score from the given input vector
$\odot$	Represents the element-wise multiplication of the corresponding position of the vector
N^l	Represents the number of nodes
$MAX(\cdot)$	Represents the maximum pooling operation
V_{fc}	Represent the feature finally input to the fully connected layer
n_k	Represents the amount of data owned by the kth
l_k	Represents the corresponding loss function
θ	Represents the parameter to be optimized
$\tilde{\theta}$	Represents the updated θ parameter
$ł_k(\cdot)$	Represents the loss function
∇	Represents the gradient
η	Represents the learning rate or step size
m	Represent the first-order moment estimation of the gradient
v	Represent the second-order moment estimation of the gradient
$\tilde{m}_t$	Represent the mean of the corrected gradient
$\tilde{v}_t$	Represent the corrected gradient has a partial variance
n_fed	Represent the number of clients in federated learning
θ_k^l	Represent the clients initialize of the local model parameters
$att(\cdot)$	Represents the calculation function of the attention mechanism
θ^l	Represents the trainable parameter of the global GCN_G of the lth layer
β_k^l	Represents the parameter confidence of the lth layer of the kth local Graph convolutional network
$\beta_k^{l,t}$	Represents the attention weight coefficient assigned to the kth participant model at time t
$\theta_G^{l,t+1}$	Represents the lth layer parameter of the global model aggregated at time $t+1$
$\theta_k^{l,t}$	Represents the parameters of the local model

Abbreviations

Abbreviations and meanings(The order sort appears in this paper)

FEDGCN	Federated Learning-Based Graph Convolutional Network
GCN	Graph Convolutional Network
FedAvg	Federated Average
Non-IID	Non-Independently Identically Distributed
GNN	Graph Neural Network
UFDA	Unsupervised Federated learning Domain Adaptation
FedMeta	Federated Meta-Learning
GraphSage	Graph Sample and Aggregate
FedProx	Federated Learning proximal term
CNN	Convolutional Neural Network
MLP	Multilayer Perceptron

References

1. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
2. Xia, M.; Wang, Z.; Lu, M.; Pan, L. MFAGCN: A new framework for identifying power grid branch parameters. *Electr. Power Syst. Res.* **2022**, *207*, 107855. [CrossRef]
3. Xia, M.; Qu, Y.; Lin, H. PANDA: Parallel asymmetric double attention network for clouds and its shadow detection. *J. Appl. Remote Sens.* **2021**, *15*, 046512. [CrossRef]
4. Song, L.; Xia, M.; Jin, J.; Qian, M.; Zhang, Y. SUACDNet: Attentional change detection network based on siamese U-shaped structure. *Int. J. Appl. Earth Obs. Geoinf.* **2021**, *105*, 102597. [CrossRef]
5. Qu, Y.; Xia, M.; Zhang, Y. Strip pooling channel spatial attention network for the segmentation of cloud and cloud shadow. *Comput. Geosci.* **2021**, *157*, 104940. [CrossRef]
6. Hu, K.; Weng, C.; Zhang, Y.; Jin, J.; Xia, Q. An Overview of Underwater Vision Enhancement: From Traditional Methods to Recent Deep Learning. *J. Mar. Sci. Eng.* **2022**, *10*, 241. [CrossRef]
7. Xu, H.; Li, J.; Xiong, H.; Lu, H. Fedmax: Enabling a highly-efficient federated learning framework. In Proceedings of the 2020 IEEE 13th International Conference on Cloud Computing (CLOUD), Beijing, China, 19–23 October 2020; pp. 426–434.
8. Huang, A.; Chen, Y.; Liu, Y.; Chen, T.; Yang, Q. Rpn: A residual pooling network for efficient federated learning. *arXiv* **2001**, arXiv:2001.08600.
9. Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konečný, J.; Mazzocchi, S.; McMahan, B.; et al. Towards federated learning at scale: System design. In Proceedings of the Machine Learning and Systems, Stanford, CA, USA, 31 March–2 April 2019; Volume 1, pp. 374–388.
10. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv* **2001**, arXiv:1603.04467.
11. Liu, Y.; Kang, Y.; Zhang, X.; Li, L.; Cheng, Y.; Chen, T.; Hong, M.; Yang, Q. A communication efficient collaborative learning framework for distributed features. *arXiv* **2001**, arXiv:1912.11187.
12. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol. (TIST)* **2019**, *10*, 12. [CrossRef]
13. Fang, W.; Xue, Q.; Shen, L.; Sheng, V.S. Survey on the Application of Deep Learning in Extreme Weather Prediction. *Atmosphere* **2021**, *12*, 661. [CrossRef]
14. Cheng, K.; Fan, T.; Jin, Y.; Liu, Y.; Chen, T.; Papadopoulos, D.; Yang, Q. Secureboost: A lossless federated learning framework. *IEEE Intell. Syst.* **2021**, *36*, 87–98. [CrossRef]
15. Liu, Y.; Kang, Y.; Xing, C.; Chen, T.; Yang, Q. A secure federated transfer learning framework. *IEEE Intell. Syst.* **2020**, *35*, 70–82. [CrossRef]
16. Peng, X.; Huang, Z.; Zhu, Y.; Saenko, K. Federated adversarial domain adaptation. *arXiv* **2019**, arXiv:1911.02054.
17. Yurochkin, M.; Agarwal, M.; Ghosh, S.; Greenewald, K.; Hoang, N.; Khazaeni, Y. Bayesian nonparametric federated learning of neural networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 7252–7261.
18. Nadiger, C.; Kumar, A.; Abdelhak, S. Federated reinforcement learning for fast personalization. In Proceedings of the 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), Sardinia, Italy, 3–5 June 2019; pp. 123–127.
19. Chen, F.; Luo, M.; Dong, Z.; Li, Z.; He, X. Federated meta-learning with fast convergence and efficient communication. *arXiv* **2018**, arXiv:1802.07876.
20. Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated optimization in heterogeneous networks. In Proceedings of the Machine Learning and Systems, Austin, TX, USA, 2–4 March 2020; Volume 2, pp. 429–450.
21. Fang, W.; Ding, Y.; Zhang, F.; Sheng, V.S. DOG: A New Background Segmentation Recognition Method based on CNN. *Neurocomputing* **2019**, *361*, 85–91. [CrossRef]
22. Fang, W.; Pang, L.; Yi, W.; Sheng, V.S. AttEF: Convolutional LSTM Encoder-Forecaster with Attention Module for Precipitation Nowcasting. *Intell. Autom. Soft Comput.* **2021**, *30*, 453–466. [CrossRef]
23. Kipf, T.N.; Welling, M. Semi-supervised classification with Graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
24. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. *arXiv* **2017**, arXiv:1706.02216.
25. Chai, D.; Wang, L.; Yang, Q. Bike flow prediction with multi-graph convolutional networks. In Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, WA, USA, 6–9 November 2018; pp. 397–400.
26. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
27. Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How powerful are graph neural networks? *arXiv* **2018**, arXiv:1810.00826.
28. Hard, A.; Rao, K.; Mathews, R.; Ramaswamy, S.; Beaufays, F.; Augenstein, S.; Eichner, H.; Kiddon, C.; Ramage, D. Federated learning for mobile keyboard prediction. *arXiv* **2018**, arXiv:1811.03604.

29. Mei, G.; Guo, Z.; Liu, S.; Pan, L. Sgnn: A graph neural network based federated learning approach by hiding structure. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 2560–2568.
30. Zhang, C.; Zhang, S.; James, J.Q.; Yu, S. FASTGNN: A Topological Information Protected Federated Learning Approach For Traffic Speed Forecasting. *IEEE Trans. Ind. Inform.* **2021**, *17*, 8464–8474. [CrossRef]
31. Lee, J.; Lee, I.; Kang, J. Self-attention graph pooling. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019.
32. Hu, K.; Li, M.; Xia, M.; Lin, H. Multi-Scale Feature Aggregation Network for Water Area Segmentation. *Remote Sens.* **2022**, *14*, 206. [CrossRef]
33. Ji, S.; Pan, S.; Long, G.; Li, X.; Jiang, J.; Huang, Z. Learning Private Neural Language Modeling with Attentive Aggregation.In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2018.
34. Chen, C.; Hu, W.; Xu, Z.; Zheng, Z. FedGL: Federated graph learning framework with global self-supervision. *arXiv* **2021**, arXiv:2105.03170.
35. Xie, H.; Ma, J.; Xiong, L.; Yang, C. Federated graph classification over non-iid graphs. *arXiv* **2021**, arXiv:2106.13423.
36. Hu, K.; Ding, Y.; Jin, J.; Weng, L.; Xia, M. Skeleton Motion Recognition Based on Multi-Scale Deep Spatio-Temporal Features. *Appl. Sci.* **2022**, *13*, 1028. [CrossRef]
37. Xia, M.; Zhang, X.; Weng, L.; Xu, Y. Multi-stage Feature Constraints Learning for Age Estimation. *IEEE Trans. Inf. Forensics Secur.* **2020**, 15, 2417–2428. [CrossRef]
38. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.