Tackling Non-IID Graphs via Decoupled Structure and Feature in Federated Graph Learning

Longwen Wang[1], Jianchun Liu[2(✉)], Xianjun Gao[2], Zhi Liu[3], and Jinyang Huang[4]

[1] School of Computer Science and Technology, Xidian University, China
abeiduoicon@gmail.com
[2] University of Science and Technology of China
jcliu17@ustc.edu.cn
[3] University of Electro-Communications, Tokyo, Japan
[4] School of Computer and Information, Hefei University of Technology, China

Abstract. Federated Graph Learning (FGL) allows clients to collaboratively train Graph Neural Networks (GNNs) without exposing their private data. Nevertheless, FGL struggles with severe non-IID data issues, leading to degraded performance of the global model, especially across diverse domains. Our analysis reveals that high data heterogeneity in FGL likely arises from notable distribution differences between graph structures and features, which significantly complicate the simultaneous alignment of the global model. Motivated by this, we propose FedDense, an efficient FGL framework that decouples the learning and sharing of structural and feature information. To better acquire structural knowledge regardless of graph features, FedDense first explicitly encodes graph structures with a separate GNN channel. The structural channel is then shared among clients, while the feature learning remains locally, ensuring that the global model reconciles only the structural knowledge, thereby reducing heterogeneity in FGL. To further facilitate knowledge acquisition efficiency of both local features and shared structures, FedDense introduces a novel Dual-Densely Connected (DDC) architecture where each layer in the local feature channel is connected to all preceding layers from its own and the shared structural channel. Extensive experiments demonstrate that FedDense with narrow layers consistently outperforms baselines, achieving higher performance while minimizing resource costs.

Keywords: Federated Graph Learning · Non-IID · Structure-Feature Decoupling.

1 Introduction

The rising interest in Graph Neural Networks (GNNs) is fueled by the extensive availability of graph data across various domains, such as chemical molecules [12], bioinformatics [11], and social networks [9]. Traditional GNNs require graph data to be centralized for processing and analysis. However, escalating privacy concerns and the increased need for cross-domain collaboration have made addressing privacy breaches and data silos crucial. To this end, Federated Graph

Learning (FGL) [15], which integrates Federated Learning (FL) [6] into the training process of GNNs, has been proposed. FGL allows clients to collaboratively train GNNs without disclosing their private data, thereby unlocking the full potential of GNNs. However, due to the severe non-IID nature of real-life graph datasets, the performance of FGL is often suboptimal [13].

Existing FGL methods predominantly rely on traditional GNNs that employ a message-passing mechanism [4], where each graph node representation is iteratively updated by aggregating features from its one-hop to multi-hop neighbors. Consequently, the client model learns both structural and feature knowledge of the local graph data and shares them among clients simultaneously in FGL. However, graph structures, as an inherent property of graphs, may embody knowledge distinct from graph features and reveal a different distribution [1]. Therefore, the primary cause of high heterogeneity in graphs compared to non-structural data may lie in the fact that, in traditional FGL methods where both structural and feature knowledge are shared and aggregated, the global model faces difficulty not only in reconciling the heterogeneity of graph features across clients but also in simultaneously balancing the differences in graph structures. This dual heterogeneity significantly increases the complexity of achieving consistency across client updates, ultimately leading to degraded performance and posing significant challenges for convergence.

To avoid this, a natural solution is to share only the structural knowledge in FGL, as graph structures (*e.g.*, graph topology and connectivity patterns) can be learned and shared independently of feature information, whereas the feature learning in graph data is based on graph structures. A straightforward approach can be a decoupled dual-channel local GNNs that separately learn structural and feature knowledge while only the structural channel is shared among clients. However, We can not overlook the fact that a simple dual-channel design may significantly increase local computational demands and model size for clients. Moreover, the basic interactions between the two channels may fail to fully exploit the potential of the dual-channel architecture and lead to insufficient utilization of shared structural knowledge in the local training, ultimately resulting in degraded learning efficiency. Hence, determining how to better leverage separate graph structures to address the non-IID issues in FGL while considering resource constraints becomes crucial.

To this end, we introduce FedDense, an efficient FGL framework that decouples the learning and sharing of structural and feature information. To ensure the acquisition and isolation of structural knowledge, FedDense introduces a structural vector alongside graph features to explicitly encode the inherent structural properties of graph data. Subsequently, FedDense utilizes decoupled dual-channel GNNs to guarantee that graph structures are learned independently of graph features. Then, only the structural channel is shared to address the complex and severe non-IID issues arising from the different distributions of graph structures and features across domains. Finally, to facilitate both local and global learning efficiency, unlike basic decoupling methods, FedDense employs an innovative Dual-Densely Connected (DDC) architecture where each

layer in the locally trained feature channel is prompted with its all preceding layers and directly benefits from the shared structural channel. The high efficiency of FedDense enables each client model to be designed with very narrow layers, achieving excellent performance with minimal local computational overhead and compact model sizes suitable for deployment. There are three key contributions of our work:

- We provide a novel analysis and perspective of the non-IID issue in FGL, highlighting the unique role and potential of inherent structural knowledge within graph data.
- We propose a new structure-feature decoupled FGL framework, FedDense, which optimizes the utilization efficiency of structural knowledge while considering resource constraints in FGL (*e.g.*, computational demand and communication cost).
- We conduct extensive experiments in four non-IID settings, demonstrating that FedDense, even with narrow layers, consistently outperforms baselines in training performance while requiring minimal resource demand.

2 Preliminaries

2.1 Graph Neural Networks (GNNs)

A typical graph $G = (V, E)$ consists of a set of nodes V and a set of edges E, where each node $v \in V$ is associated with a feature vector $\mathbf{x}_v$. We denote the representation of node v as $\mathbf{h}_v$, and it can be iteratively updated by aggregating the representations of its one-hop neighbors $\mathcal{N}(v)$ as:

$$\mathbf{m}_v^{(\ell)} = \text{AGGREGATE}\left(\left\{\mathbf{h}_u^{(\ell-1)} | u \in \mathcal{N}(v)\right\}\right), \tag{1}$$

$$\mathbf{h}_v^{(\ell)} = \text{UPDATE}\left(\mathbf{h}_v^{(\ell-1)}, \mathbf{m}_v^{(\ell)}\right), \tag{2}$$

where $\mathbf{h}_v^{(\ell)}$ is the updated representation of the node v at the ℓ-th layer. Different AGGREGATE and UPDATE functions allow for the implementation of various types of GNNs with distinct focuses. GNNs can be applied to various tasks, such as node classification, link prediction, and graph classification. In this paper, we focus primarily on graph classification, where GNNs combine the representations of all nodes to form a graph-level representation $\mathbf{h}_G$ using pooling methods such as average, sum, or max pooling.

2.2 Federated Graph Learning (FGL)

A typical FGL system consists of a Parameter Server (PS) and a set of N clients that collaboratively train a global GNN model. Each client i holds a private graph dataset d_i, and the total samples across all clients are denoted as D. The training process of FGL is divided into T rounds. At the start of each training

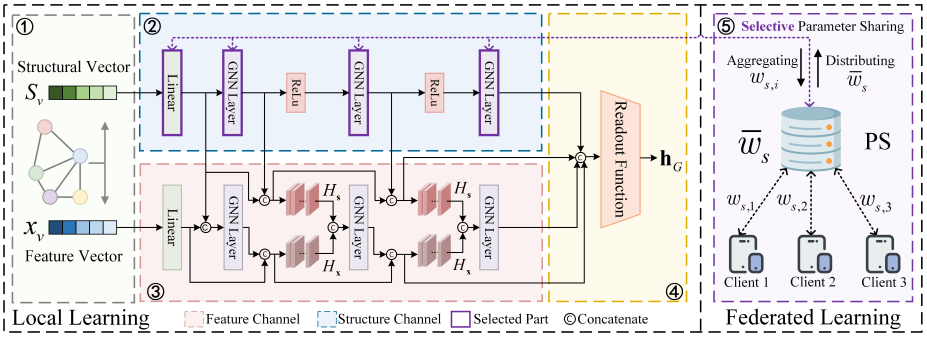


Fig. 1: An overview of the proposed FedDense framework with 3 GNN layers as an example. The left box represents the local training process with structural patterns decoupling and DDC architecture of each client. The right box illustrates the global structure-based federated sharing scheme.

round $t \in \{1, \ldots, T\}$, the PS distributes the global model parameters $\bar{w}^{(t)}$ to all clients. Upon receiving $\bar{w}^{(t)}$, each client i performs local training on its private graph data d_i and uploads the updated model parameters $w_i^{(t)}$ back to the PS. At the end of round t, the PS aggregates these updates for the next round. The typical aggregation method used in FGL is FedAvg [8], which averages the model updates from all clients by:

$$\bar{w}^{(t+1)} = \sum_{i=1}^{N} \frac{|d_i|}{|D|} w_i^{(t)}, \tag{3}$$

where $|d_i|$ denotes the size of data samples of client i and $|D|$ represents the total size of samples over all clients.

The global model optimization in FGL aims to minimize the overall loss across all participating clients, denoted as:

$$\underset{(w_1, w_2, \cdots, w_i)}{\arg\min} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i\left(w_i\right), \tag{4}$$

where $\mathcal{L}_i(\cdot)$ and w_i are the loss function and model parameters of client i, respectively. However, due to the prevalence of non-IID data in real-life graph datasets, the performance of FGL is often suboptimal.

3 Methodology

In this section, we detail our proposed FedDense framework, which is illustrated in Fig. 1. FedDense focuses on better decoupling the graph structures and features at two levels: the data level and the feature map level. At the data level,

FedDense introduces a structural vector alongside node features (①) to explicitly capture the unique structural patterns inherent in graph data itself. At the feature map level, we propose a Dual-Densely Connected GNN architecture. Specifically, we first employ dual-channel (② and ③) GNNs to separately learn feature and structural knowledge with the decoupled vectors. Additionally, FedDense establishes dense connections between the dual-channel GNNs by integrating all preceding feature maps at each layer in the feature channel (③) and aggregates the collective knowledge within all hidden layers to generate the final graph-level representation (④). Finally, we design a structure-based parameter sharing scheme where only the structural channel is shared among clients (⑤).

3.1 Structural Patterns Decoupling

Existing GNNs primarily rely on feature information for message-passing. However, although GNNs implicitly incorporate structural knowledge through the message-passing mechanism by iteratively aggregating one-hop neighboring node features, this approach diminishes the direct learning of unique topological structures (*e.g.*, node degree) in graph data. However, inherent structural patterns of graph data carry significant and distinctive information alongside graph features, especially in cross-domain scenarios. Therefore, it is crucial to explicitly leverage structural knowledge and learn from both feature and structural information. To this end, inspired by [3, 13], we introduce a structural vector into each graph node. Specifically, in addition to the node's feature vector $\mathbf{x}_v$, we extract and convert the node's inherent topological information into a structural vector $\mathbf{s}_v$, defined as:

$$\mathbf{s}_v = f\left([s_1, s_2, s_3, \ldots, s_n]\right), \tag{5}$$

where $[s_1, s_2, s_3, \ldots, s_n]$ are structural information encodings. To capture the local/global structural patterns of graph data, potential options include one-hot degree vectors, random walk transition matrices, and positional embeddings [2, 3]. The function $f(\cdot)$ serves as a fusion function, which can be implemented using techniques such as concatenation, fully connected layers, or pooling [7]. Notably, both the structural encodings $[s_1, s_2, s_3, \ldots, s_n]$ and the function $f(\cdot)$ can be tailored according to the specific task, highlighting the versatility and adaptability of this approach. By incorporating the structural vector to each node, an additional dimension of graph data is added, making the node representations more robust and informative, denoted as $\{\mathbf{x}_v, \mathbf{s}_v\}$.

3.2 Dual-Densely Connected Architecture

In order to separately capture and process feature and structural information in graph data, the basic decoupled dual-channel GNN architecture is a good starting point. However, it has limitations. The straightforward addition of extra channels risks imposing significant local computation demands, especially for distributed paradigms like FGL, indicating the need for further exploration of more efficient decoupling mechanisms.

A traditional GNN layer can be viewed as an aggregation from one-hop neighbors. Therefore, each layer of stacked GNNs is able to fetch different scales of local or global feature and structure insights within multi-hops, indicating that the feature maps in GNNs are highly informative, especially in a decoupled dual-channel design. Inspired by this, we propose a novel Dual-Densely Connected (DDC) architecture. Specifically, we first employ dual-channel GNNs to separately learn feature and structural information with decoupled feature vector $\mathbf{x}_v$ and structural vector $\mathbf{s}_v$ at the data level. Additionally, we establish dense connections between the dual channels at the feature map level, where each layer in the feature channel receives additional inputs from the outputs of all preceding layers in both channels. This dual-dense connectivity allows the multi-scale insights of feature maps in both channels to be collectively leveraged in the feature channel.

Initialization. The DDC architecture employs two parallel GNNs: one for feature learning and one for structural learning. Both channels start with a linear initialization layer. In the feature channel, the feature vector $\mathbf{x}_v$ of each node v is processed by a linear layer, transforming it into a hidden representation $\mathbf{x}_v^{(0)}$. Simultaneously, in the structural channel, the corresponding structural vector $\mathbf{s}_v$ is passed through a separate linear layer, producing the representation $\mathbf{s}_v^{(0)}$ with the same dimension as $\mathbf{x}_v^{(0)}$.

Dual-Dense Connectivity. After initialization, both channels follow L stacked GNN layers. To maintain simplicity and preserve the integrity of structural information, the input to the ℓ-th GNN layer in the structural channel is directly derived from the output of the previous layer $\mathbf{s}_v^{(\ell-1)}$ in this channel, where $\ell \in \{1, \cdots, L\}$. Meanwhile, to enhance internal interactions within the decoupled networks and leverage multi-scale information from their stacked hidden layers, each layer in the feature channel receives the feature maps of all preceding layers from both channels as input: $\mathbf{c}_v^{(\ell)} = \text{Concat}[\boldsymbol{\alpha}_v^{(\ell)}, \boldsymbol{\beta}_v^{(\ell)}]$, where Concat $[\cdot]$ denotes the concatenation operation. The $\boldsymbol{\alpha}_v^{(\ell)}$ and $\boldsymbol{\beta}_v^{(\ell)}$ can be denoted as:

$$\boldsymbol{\alpha}_v^{(\ell)} = H_{\mathbf{x}}(\text{Concat}[\mathbf{x}_v^{(0)}, \mathbf{x}_v^{(1)}, \cdots, \mathbf{x}_v^{(\ell-1)}]), \tag{6}$$

$$\boldsymbol{\beta}_v^{(\ell)} = H_{\mathbf{s}}(\text{Concat}[\mathbf{s}_v^{(0)}, \mathbf{s}_v^{(1)}, \cdots, \mathbf{s}_v^{(\ell-1)}]), \tag{7}$$

where $\mathbf{x}_v^{(0)}, \mathbf{x}_v^{(1)}, \cdots, \mathbf{x}_v^{(\ell-1)}$ and $\mathbf{s}_v^{(0)}, \mathbf{s}_v^{(1)}, \cdots, \mathbf{s}_v^{(\ell-1)}$ refer to the feature maps produced in layers 0 through $\ell - 1$ from the feature and structural channels, respectively. We define $H_{\mathbf{x}}(\cdot)$ and $H_{\mathbf{s}}(\cdot)$ as non-linear transformations between hidden GNN layers in the feature channel, consisting of a composite function of operations such as Batch Normalization (BN), Dropout, rectified linear units (ReLU), and Pooling. For the final graph-level embedding $\mathbf{h}_G$, rather than relying solely on the output of the final layer, we consider and concatenate the feature maps of all the hidden layer outputs generated across both channels. The concatenated representation is then transformed into the graph-level embedding via

a readout function. With the DDC architecture, FedDense not only ensures the independent learning of structural knowledge with two parallel channels but also guarantees the different insights they learn individually are fully integrated and leveraged with dual-dense connectivity. Additionally, by combining all feature maps throughout the network to generate the final graph embeddings, FedDense considers the collective knowledge across both channels, thus achieving robust and comprehensive learning of graph data.

3.3 Structure-based Federated Sharing

In FGL, the data held by each client is typically heterogeneous and this issue is particularly pronounced in graph data. Considering GNNs inherently learn feature information based on graph structures, a feature-based sharing strategy inevitably causes the global aggregation to incorporate both structural and feature information, making it difficult for the global model to adapt to heterogeneity caused by both graph structures and features and ultimately leading to degraded performance. Therefore, unlike traditional FGL methods, where all model parameters are shared among clients, FedDense restricts parameter sharing to the structural parameters only, specifically the learnable parameters of each layer in the structural channel. Hence, Eq. (3) can be reformulated as $\bar{w}_s^{(t+1)} = \sum_{i=1}^{N} \frac{|d_i|}{|D|} w_{s,i}^{(t)}$, where $\bar{w}_s^{(t+1)}$ represents the aggregated structural parameters at the PS, and $w_{s,i}^{(t)}$ denotes the updated structural parameters of client i in round t. The feature parameters are neither shared nor updated through federated learning but are instead optimized locally within each client. Notably, with the proposed DDC architecture, even though feature learning is conducted locally, it can still directly benefit from the shared graph structures, enhancing the efficiency and effectiveness of the local feature learning.

4 Experiments

4.1 Datasets and Experimental Setup

Datasets. We utilize a total of 15 public graph classification datasets [10] across 4 different domains: seven Molecules datasets (MUTAG, BZR, COX2, DHFR, PTC-MR, AIDS, NCI1), two Bioinformatics datasets (ENZYMES, DD), three Social Networks datasets (COLLAB, IMDB-BINARY, IMDB-MULTI), and three Computer Vision datasets (Letter-low, Letter-high, Letter-med). To simulate data heterogeneity in FGL, we define four non-IID settings: (1) Single-domain (Single) with only Molecules; (2) Cross-domain (Cross-Sim) with similar domains (Molecules and Bioinformatics); (3) Cross-domain (Cross-Diff) with different domains (Bioinformatics, Social Networks, and Computer Vision); and (4) Multi-domain (Multi) combining all domains. In each setting, the graph data for each client is derived from one of the corresponding datasets and is randomly split into a ratio of 8:1:1 for training, validation, and testing.

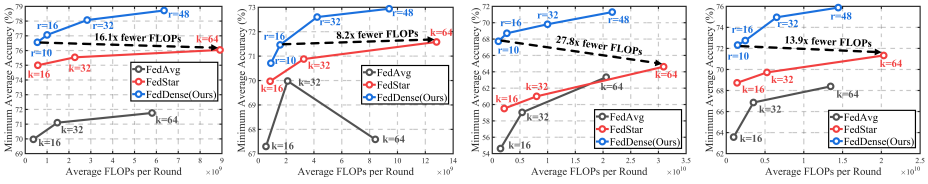
Table 1: Performance in four non-IID settings. We present the average accuracy and gain over Local for all FGL methods. The best performances in each setting are **bold**, while the second-best performances are underlined.

Settings	Single		Cross-Sim		Cross-Diff		Multi	
# datasets	7		9		8		15	
Accuracy	avg	gain	avg	gain	avg	gain	avg	gain
Local	75.50±1.77	-	72.12±1.81	-	67.06±1.84	-	71.12±1.14	-
FedAvg	75.07±3.72	-0.43	71.73±1.01	-0.39	64.80±2.60	-2.26	68.67±0.06	-2.45
FedProx	74.00±2.03	-1.50	72.57±1.82	0.45	63.94±3.64	-3.12	69.18±0.08	-1.94
GCFL	75.36±1.17	-0.14	72.98±1.37	0.86	64.54±1.88	-2.25	70.32±0.10	-0.8
FedStar	79.32±2.47	3.82	74.42±2.37	2.30	67.48±3.04	0.42	73.70±1.54	2.58
FedDense(r=16)	79.86±2.62	4.36	74.51±2.35	2.39	71.66±2.03	4.60	74.35±1.85	3.23
FedDense(r=32)	**80.16**±2.74	**4.66**	**75.74**±2.44	**3.62**	**72.44**±2.05	**5.38**	**75.31**±1.68	**4.19**

Baselines and Training Details. We employ five baselines in our experiments: (1) Local, where each client trains its model locally; (2) FedAvg [8], a standard FGL approach; (3) FedProx [5], which was proposed to handle system and statistical heterogeneity in FL; (4) GCFL [14], which tackles non-IID graph data through a dynamic clustering technique; and (5) FedStar [13], a state-of-the-art FGL framework that addresses the non-IID issues with structure-based domain-invariant knowledge. We set the hidden size to 64 for all baselines. For FedDense, the hidden size is controlled by the hyperparameter r. The local epoch is set to 1, and the number of communication rounds is 200 for all FGL methods. To ensure comparability, both FedStar and FedDense use the same initial structural encodings: a degree-based embedding and a random walk-based positional embedding [3], both with dimensions of 16. All experiments are conducted on one NVIDIA GeForce RTX 4090 GPU and run for five random repetitions.

4.2 Experimental Results.

Accuracy Performance. As shown in Table 1, FedDense surpasses all competing baselines in four non-IID settings. In Cross-Diff and Multi settings, where the data across clients is more heterogeneous, all baselines exhibit severe performance degradation, with most methods failing to surpass the Local baseline. However, under these highly heterogeneous conditions, FedDense ($r = 32$) achieves impressive average accuracy gains of 5.38% and 4.19%, respectively, significantly outperforming the existing state-of-the-art FedStar by notable margins of 4.98% and 1.61%. Remarkably, even with a small r (*i.e.*,$r = 16$), our framework still achieves excellent performance gain(*i.e.*, 4.60% and 3.23%) and continues to



(a) Single (b) Cross-Sim (c) Cross-Diff (d) Mulit

Fig. 2: Computation efficiency in four non-IID settings, presenting the minimum average accuracy across five random repetitions and the average FLOPs per client per round for each FGL method. r and k denote the hidden size of the local GNNs in FedDense and other FGL methods, respectively.

surpass FedStar (*i.e.*, 4.18% and 0.65%). The superior performance of FedDense can be attributed to its structure-based sharing scheme and DDC architecture. The purity of the shared graph structures, coupled with the local integration of personal features and global structural insights, significantly enhances knowledge acquisition across clients, enabling FedDense to effectively model complex and diverse knowledge in both local and cross-domain graphs.

Computational Efficiency. One of the primary advantages of our proposed framework is its remarkable computational efficiency. As illustrated in Fig. 2, FedDense significantly outperforms FedAvg and FedStar in minimum average accuracy across five random repetitions while requiring minimal local computation. Although FedStar achieves better accuracy compared to FedAvg, its significant local computation demands make it less suitable for distributed paradigms like FGL. In contrast, FedDense stands out for its exceptional efficiency, achieving the best accuracy performance while demanding minimal computational resources in all non-IID settings. Notably, in the highly heterogeneous Cross-Diff setting, FedDense ($r = 10$) surpasses FedStar ($k = 64$) by a large margin in accuracy while requiring 27.8 times lower FLOPs per client per round, demonstrating that extracting knowledge from feature maps in both the personal feature channel and the shared structural channel significantly facilitates knowledge acquisition in FGL. Despite constraints on computational resources and limited client participation time, FedDense delivers outstanding performance.

5 Conclusion

To avoid the severe heterogeneity caused by different distributions of graph structures and features, we decouple graph structures alongside graph features and devise a structure-based parameter sharing strategy in federated graph learning. Moreover, We employ a dual-densely connected architecture to facilitate both local and global knowledge acquisition. The extensive experimental results demonstrate that FedDense with narrow layers achieves state-of-the-art performance with minimum resource demands.

Acknowledgments.

This article is supported in part by the Jiangsu Province Science Foundation for Youths (Grant No. BK20230275); in part by the Anhui Province Science Foundation for Youths (Grant No. 2408085QF185); in part by USTC Research Funds of the Double First-Class Initiative (Grants No. WK2150110033).

References

1. Buffelli, D., Vandin, F.: The impact of global structural information in graph neural networks applications. Data **7**(1), 10 (2022)
2. Cui, H., Lu, Z., Li, P., Yang, C.: On positional and structural node features for graph neural networks on non-attributed graphs. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management (2022)
3. Dwivedi, V.P., Luu, A.T., Laurent, T., Bengio, Y., Bresson, X.: Graph neural networks with learnable structural and positional representations. In: International Conference on Learning Representations (2022)
4. Huang, J., Liu, B., Miao, C., Zhang, X., Liu, J., Su, L., Liu, Z., Gu, Y.: Phyfinatt: An undetectable attack framework against phy layer fingerprint-based wifi authentication. IEEE Transactions on Mobile Computing (2023)
5. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. In: Dhillon, I., Papailiopoulos, D., Sze, V. (eds.) Proceedings of Machine Learning and Systems. vol. 2, pp. 429–450 (2020)
6. Liu, J., Yan, J., Xu, H., Wang, Z., Huang, J., Xu, Y.: Finch: Enhancing federated learning with hierarchical neural architecture search. IEEE Transactions on Mobile Computing (2023)
7. Ma, C., Mu, X., Sha, D.: Multi-layers feature fusion of convolutional neural network for scene classification of remote sensing. IEEE Access **7**, 121685–121694 (2019)
8. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017)
9. Monti, F., Frasca, F., Eynard, D., Mannion, D., Bronstein, M.M.: Fake news detection on social media using geometric deep learning. arXiv preprint arXiv:1902.06673 (2019)
10. Morris, C., Kriege, N.M., Bause, F., Kersting, K., Mutzel, P., Neumann, M.: Tudataset: A collection of benchmark datasets for learning with graphs. arXiv preprint arXiv:2007.08663 (2020)
11. Muzio, G., O'Bray, L., Borgwardt, K.: Biological network analysis with deep learning. Briefings in bioinformatics **22**(2), 1515–1530 (2021)
12. Qian, C., Xiong, Y., Chen, X.: Directed graph attention neural network utilizing 3d coordinates for molecular property prediction. Computational Materials Science **200**, 110761 (2021)
13. Tan, Y., Liu, Y., Long, G., Jiang, J., Lu, Q., Zhang, C.: Federated learning on non-iid graphs via structural knowledge sharing. In: AAAI (2023)
14. Xie, H., Ma, J., Xiong, L., Yang, C.: Federated graph classification over non-iid graphs. Advances in neural information processing systems **34**, 18839–18852 (2021)
15. Zhang, H., Shen, T., Wu, F., Yin, M., Yang, H., Wu, C.: Federated graph learning–a position paper. arXiv preprint arXiv:2105.11099 (2021)