Historical Embedding-Guided Efficient Large-Scale Federated Graph Learning

ANRAN LI*, Nanyang Technological University, Singapore
YUANYUAN CHEN*, Nanyang Technological University, Singapore
JIAN ZHANG†, Nanyang Technological University, Singapore
MINGFEI CHENG, Singapore Management University, Singapore
YIHAO HUANG, Nanyang Technological University, Singapore
YUEMING WU, Nanyang Technological University, Singapore
ANH TUAN LUU, Nanyang Technological University, Singapore
HAN YU†, Nanyang Technological University, Singapore

Graph convolutional networks (GCNs) are promising for graph learning tasks. For privacy-preserving graph learning tasks involving distributed graph datasets, federated learning (FL)-based GCN (FedGCN) training is required. An important open challenge for FedGCN is scaling to large graphs, which typically incurs 1) high computation overhead for handling the explosively-increasing number of neighbors, and 2) high communication overhead of training GCNs involving multiple FL clients. Thus, neighbor sampling is being studied to enhance the scalability of FedGCNs. Existing FedGCN training techniques with neighbor sampling often produce extremely large communication and computation overhead and inaccurate node embeddings, leading to poor model performance. To bridge this gap, we propose the Federated Adaptive Attention-based Sampling (FedAAS) approach. It achieves substantial cost savings by efficiently leveraging historical embedding estimators and focusing the limited communication resources on transmitting the most influential neighbor node embeddings across FL clients. We further design an adaptive embedding synchronization scheme to optimize the efficiency and accuracy of FedAAS on large-scale datasets. Theoretical analysis shows that the approximation error induced by the staleness of historical embedding is upper bounded, and the model is guaranteed to converge in an efficient manner. Extensive experimental evaluation [1] against four state-of-the-art baselines on six real-world graph datasets show that FedAAS achieves up to 5.12% higher test accuracy, while saving communication and computation costs by 95.11% and 94.76%, respectively.

CCS Concepts: • **Computing methodologies** → **Distributed algorithms**; • **Information systems** → *Data mining*.

Additional Key Words and Phrases: Graph federated learning, graph sampling, historical embedding

*Equal contribution.
†Jian Zhang and Han Yu are corresponding authors.
[1]Our code is released at https://github.com/cyyever/distributed_learning_simulator/tree/fed_aas.

Authors' addresses: Anran Li, anran.li@ntu.edu.sg, Nanyang Technological University, Singapore; Yuanyuan Chen, yuanyuan.chen@ntu.edu.sg, Nanyang Technological University, Singapore; Jian Zhang, jian_zhang@ntu.edu.sg, Nanyang Technological University, Singapore; Mingfei Cheng, snowbirds.mf@gmail.com, Singapore Management University, Singapore; Yihao Huang, huangyihao22@gmail.com, Nanyang Technological University, Singapore; Yueming Wu, wuyueming21@gmail.com, Nanyang Technological University, Singapore; Anh Tuan Luu, anhtuan.luu@ntu.edu.sg, Nanyang Technological University, Singapore; Han Yu, han.yu@ntu.edu.sg, Nanyang Technological University, Singapore.

ACM 2836-6573/2024/6-ART144
https://doi.org/10.1145/3654947

ACM Reference Format:
Anran Li, Yuanyuan Chen, Jian Zhang, Mingfei Cheng, Yihao Huang, Yueming Wu, Anh Tuan Luu, and Han Yu. 2024. Historical Embedding-Guided Efficient Large-Scale Federated Graph Learning. *Proc. ACM Manag. Data* 2, 3 (SIGMOD), Article 144 (June 2024), 24 pages. https://doi.org/10.1145/3654947

1 INTRODUCTION

Graph convolutional networks (GCNs) [23] are powerful models that learn representations from complex graph-structured data. By stacking multiple graph convolution layers, GCNs can learn node representations by aggregating information from distant neighbors. GCNs and their variants [9, 16, 32] have been applied in various domains, *e.g.*, social network predictions [10, 24], drug discovery [33, 36], knowledge graph mining [8, 40] and traffic flow modeling [39]. Due to privacy concerns, it is common for large-scale graph data owned by different entities to be located on multiple decentralized sites. For example, pharmaceutical companies would significantly benefit from graph data from other healthcare providers, but they cannot afford to disclose their sensitive private data. With heterogeneous subgraphs separately stored by data owners (*e.g.*, molecule graphs owned by pharmaceutical companies and social networks located in the social app of end users), building globally applicable GCNs requires collaboration.

Federated learning (FL) is a collaborative machine learning paradigm that performs distributed training of models on local datasets owned by FL clients (*e.g.*, sensors, edge devices) [17, 25, 29]. The global model can be obtained by aggregating a large corpus of clients' local models without exposing their local data to any third party. Due to the advantages of FL, federated graph learning (FedGL) [1, 18, 37] for collaborative GCN training while preserving data privacy and reducing bandwidth has gained traction. Based on the distribution of graph data, there are two main categories of FedGL: 1) inter-graph FedGL, and 2) intra-graph FedGL. Under inter-graph FedGL [18, 37], all local data samples are graph data, and the global model performs graph-level task. Under intra-graph FedGL [1, 7, 30], each client owns a part of the graph or one correlated graph, and global model performs node-level or link-level tasks. Intra-graph FL is common in practice, *e.g.*, in online social application where each user has a local social network, and all networks constitute the latent entire human social network. The developers are able to design friend recommendation algorithms based on intra-graph FL to avoid violating users' social privacy [44].

The key challenge hindering the wide adoption of intra-graph FedGL is scaling it to large graphs (*e.g.*, a social network maintained by Facebook contains over three billion users, and the corresponding graph data size may be several hundreds of gigabytes). Firstly, the exponentially increasing dependency of neighboring nodes over layers (*i.e.*, neighbor explosion) causes the computation graph to be extremely large, which can exceed FL clients' local storage capacity. Secondly, intra-graph FedGL requires intermediate embedding communication across clients. There can be edges between nodes that are stored by different clients. As embedding calculation for one node requires information from its recursive neighbors several hops away, which may be stored on other devices, fetching neighbor information from other clients incurs high communication overhead. Treating sub-graphs at different devices as independent, thereby ignoring the information from neighbors across clients, results in high prediction errors or slow model convergence [1, 7].

To alleviate these challenges, we focus on the graph sampling approach which adaptively transmits important cross-client neighbor embeddings for large-scale distributed graphs. There are a number of existing graph sampling methods for efficient graph learning under centralized learning settings instead of FL settings: 1) node-wise sampling, and 2) layer-wise sampling. Node-wise sampling methods [3, 5, 16] iteratively sample a number of neighbors for each node based on specific probabilities (*e.g.*, calculated based on node centrality [41]). However, the number of sampling nodes may grow exponentially as more layers are constructed. Besides, the shared

neighbors of some nodes can result in a large number of embedding computation redundancy, which incurs unnecessarily large computation overhead [20]. Layer-wise sampling methods [2, 47] select a number of nodes for each GCN layer. Since multiple nodes are jointly sampled in each layer, the time cost of the sampling process is significantly reduced by avoiding the exponential extension of neighbors. However, since nodes of different layers are sampled independently, some sampled nodes might not be connected with the ones in the previous layer, which causes the embeddings of some unconnected nodes to be lost during the graph convolution operation, thereby deteriorating model performance [1, 2]. Besides, since all those methods are designed for centralized scenarios, they neglect communication cost and incur substantial communication burden when the volume of the training data is large, and raise serious privacy concerns.

As for graph sampling in FL, in [1], graph sampling is conducted with the node-wise sampling method by utilizing deep reinforcement learning (DRL). However, since the proposed DRL reward function only consists of learning speed and accuracy, while ignoring the impact of the topological structure of the graph, it leads to inaccurate node embeddings and thus suboptimal model performance. Besides, it incurs large computational overhead since each FL client needs to train two DRL networks to calculate the neighbor node sampling probability. Another work [7] proposed to determine the optimal neighbor sampling interval by solving the online problem to achieve a trade-off between model convergence and time cost. However, it ignores the influence of different neighboring nodes and directly adopts a random sampling strategy, which introduces unnecessarily large communication costs as well as bias and variance in model gradients.

To address these limitations of existing work, we propose a novel federated graph sampling scheme - the federated adaptive attention-based sampling (FedAAS) approach - for large-scale graph data in node classification tasks. It achieves substantial cost savings by efficiently leveraging historical embedding estimators and focusing the limited communication and computation resources on transmitting and aggregating the most influential cross-client neighboring node embeddings. By designing an adaptive embedding synchronization scheme, it is further capable of achieving the best error-runtime trade-off between model convergence and time cost. The key advantages of FedAAS are summarized as follows:

- **Scalability:** FedAAS is able to scale FedGL to large graphs with constant memory consumption with respect to input node sizes. For a given set of within-client nodes, FedAAS prunes the GCN computation graph so that only nodes inside the current clients and their direct 1-hop cross-client neighbors are retained, regardless of GCN depth. Historical embeddings are used to accurately fill in the inter-dependency information of cross-client neighbors.
- **Efficiency:** FedAAS achieves highly efficient federated graph learning by prioritizing highly influential neighbors and adaptive embedding transmission based on historical embeddings. It reduces unnecessary cross-client neighbor embedding communications through the adaptive attention-based sampling to determine the optimal communication period and achieves faster convergence with lower computation and communication costs.
- **Interpretability:** The adaptive attention-based rules guiding the FedAAS decision-making process with regard to updating neighbor selection intervals are transparent. FedAAS starts with infrequent embedding synchronization to improve convergence speed, and then increases the frequency to gradually reduce the variance of gradients and construct a global FedGCN model with low prediction errors.

We evaluated FedAAS on six graph datasets of different scales with real-world workloads. Compared to the four state-of-the-art approaches, FedAAS achieves significant cost savings in training high-performance FedGL models with thousands of data owners. On average, it improves test accuracy by 5.12%, while incurring 95.11% and 94.76% lower computation and lower communication cost,

respectively. In this way, FedAAS achieves the best efficiency and accuracy on several large-scale graph benchmark datasets. Furthermore, it enables the application of expressive and hard-to-scale-up models on large graphs, leading to state-of-the-art results on several large-scale graph benchmark datasets.

2 RELATED WORK

To reduce the overhead incurred for training GCN models involving large graphs, graph sampling methods have been proposed, mostly under cenntralized learning settings [5, 16, 28], albeit techniques designed for FL settings [27, 31, 45] are starting to emerge.

2.1 Graph Sampling under Centralized Learning Settings

Graph sampling methods in centralized learning can be mainly divided into two main categories: 1) node-wise sampling, and 2) layer-wise sampling. Node-wise sampling [3, 5, 16] iteratively samples a number of neighbors for each node based on specific probabilities (*e.g.*, calculated based on node centrality [41]). A representative work under this category is GraphSAGE [16]. It randomly selects a fixed number of neighbors for each node in each graph network layer. However, the number of sampling nodes might grow exponentially as more layers are constructed, a phenomenon referred to as *neighbor explosion* [13]. Besides, Huang et al. [20] have pointed out that the shared neighbors of some nodes results in a large number of embedding computation redundancies, thereby incurring high computation overhead unnecessarily. Several recent works, such as VR-GCN [3] and Cluster-GCN [4], have been proposed to improve the performance of node-wise sampling approaches. Nevertheless, they are still unable to address this problem.

Instead of sampling neighbors for each node, layer-wise sampling methods (*e.g.*, FastGCN [2], LADIES [47]) sample a number of nodes for each GCN layer. For example, FastGCN [2] interpret graph convolutions as integral transforms of embedding functions under probability measures. Such an interpretation allows for the use of Monte Carlo approaches to consistently estimate the integrals, and samples a certain number of nodes in each layer independently based on the importance sampling [21]. Since multiple nodes are jointly sampled in each layer, the time cost of the sampling process is significantly reduced by avoiding the exponential extension of neighbors. However, since nodes of different layers are sampled independently, some sampled nodes may have no connections with the ones in the previous layer. Then, the embeddings of some unlinked nodes may be lost during graph convolution operations, which can result in performance deterioration. Besides, since all those methods are designed for centralized scenarios, they neglect communication cost and incur substantial communication burden when the volume of the training data is large, and raise serious privacy concerns. In brief, such methods are not directly applicable to FL systems, where the local training data are private, and the majority of clients are resource-constrained.

2.2 Graph Sampling under FL Settings

Relatively few works on graph sampling under FL settings have been proposed for the node classification tasks [1, 7, 46], In [1], graph sampling is conducted with the node-wise sampling method by utilizing deep reinforcement learning (DRL). However, since the proposed reward function of DRL only covers learning speed and accuracy, while ignoring the graph topology. This leads to inaccurate node embeddings and thus, inferior model performance. In addition, since each local client needs to train two additional deep networks separately to calculate the selection probabilities, it would incur large computation overhead. In [7], a technique for determining the optimal neighbor sampling interval by solving the online problem to achieve a trade-off between convergence and time cost has been proposed. However, since it ignores the influence of different neighboring nodes and directly adopts a random sampling strategy, it incurs high communication

cost unnecessarily, and results in large bias and variance in the gradients. The work [46] directly applies GraphSAGE [16] in FL, which also suffers from the neighbor explosion problem. The proposed FedAAS aims to address these limitations facing existing FL graph sampling techniques.

3 PROBLEM FORMULATION AND MAIN IDEA

3.1 Preliminaries on Federated Graph Learning

There are two types of entities involved in a typical FedGL scenario: an FL server S, and K distributed FL clients $\{1, 2, \cdots, K\}$. Each client k possesses a local dataset $D_k = (G_k, Y_k)$, where $G_k = (V_k, E_k)$ is an undirected graph in D_k with $|V_k|$ vertices and $|E_k|$ edges. We assume that each client is aware of the existence of neighboring nodes maintained by other clients, but cannot directly access their feature vectors. This is consistent with real-world graph data distribution [44]. Taking a social network as an example, each user only maintains his friend list locally, while the details about his friends are contained in their friends' profiles.

We focus on the task of node classification, where each vertex $v \in V_k$ is associated with a feature vector $x_v \in X_k$ and a label $y_v \in Y_k$. Under the coordination of the FL server, all FL clients collaboratively train a global FedGCN model $\theta^* \in \mathbb{R}^d$ by sharing their respective local models updated with their respective private datasets, and obtaining the representation h_{v,θ^*}, from which y_v can be predicted. Generally, FedGCN follows a *neural message passing mechanism* [15]. For client k, the $(l+1)$-th layer is defined as:

$$h_{v,\theta}^{(l+1)} = \gamma_\theta^{(l+1)}\left(h_{v,\theta}^{(l)}, \{h_{w,\theta}^{(l)}\}_{w\in N(v)}\right) = A_\theta^{(l+1)}\left(h_{v,\theta}^{(l)}, U_\theta^{(l+1)}(h_v^{(l)}, \{h_{w,\theta}^{(l)}\}_{w\in N(v)})\right). \tag{1}$$

Here, $h_{v,\theta}^{(l)}$ is the embedding of client k's node v in layer l. $h_{v,\theta}^{(1)} = x_v$. $N(v)$ denotes the set of neighbor nodes of v. For simplicity, we use $h_v^{(l)}$ instead of $h_{v,\theta}^{(l)}$ in the rest of the paper.

Here, the function $\gamma_\theta^{(l+1)}$ operates on multiple sets. It can be decomposed into two components: 1) aggregation function $A_\theta^{(l+1)}$, which takes the embeddings of node v and its neighbors as input, and outputs the aggregated neighborhood embedding; and 2) updating function $U_\theta^{(l+1)}$, which combines the embedding of v and the aggregated neighborhood embedding to update the embedding of node v for the next layer. The functions $A_\theta^{(l+1)}$ and $U_\theta^{(l+1)}$ are parameterized by θ. Consequently, we formulate FedGCN as a distributed optimization problem to minimize the aggregated risk:

$$\theta^* = \arg\min\{F(h^{(L)}, \theta) = \sum_{k=1}^{K} \frac{|V_k|}{V} F_k(h^{(L)}, \theta)\}, \text{where } F_k(h^{(L)}, \theta) = \frac{1}{|V_k|}\sum_{v\in V_k} f(h_v^{(L)}, \theta, y_v) \tag{2}$$

where $h_v^{(L)}$ is the embedding of node v from the last layer L, and can be calculated by following Eq. (1). $f(h_v^{(L)}, \theta, y_v)$ and $F_k(h^{(L)}, \theta)$ represent loss functions of an individual sample x_v on client k's local model and all samples on client k's local model. $F(h^{(L)}, \theta)$ represents loss function of the global model, and $V = \sum_{k=1}^{K} |V_k|$. Similarly, we simplify the notations of $f(h_v^{(L)}, \theta, y_v)$, $F_k(h^{(L)}, \theta)$ and $F(h^{(L)}, \theta)$ to $f(h_v^{(L)})$, $F_k(h^{(L)})$ and $F(h^{(L)})$, respectively. We denote the *receptive field* of a node $v \in V_k$ as all its L-hop neighbors (*i.e.*, nodes that are reachable from v within L hops).

3.2 Problem Formulation

To solve the optimization problem in Eq. (2), existing works use mini-batch stochastic gradient decent (SGD) optimization instead of full-gradient over all labeled nodes. For the general message

scheme given in Eq. (1), the execution can be formulated as:

$$h_v^{(l+1)} = \gamma_\theta^{(l+1)}\Big(h_v^{(l)}, \underbrace{\{h_w^{(l)}\}_{w\in N(v)\cap V_k}}_{\text{Within-client nodes}} \cup \underbrace{\{h_w^{(l)}\}_{w\in N(v)\setminus V_k}}_{\text{Cross-client nodes}}\Big). \tag{3}$$

Here, we separate the neighbor node information of node $v \in V_k$ into two parts: 1) the local information of the neighbor nodes $N(v)$ which are part of the current client k, and 2) the information of neighbor nodes which are not included in the current client but are part of the cross-client. We denote $[K] := \{1, 2, \cdots, K\}$ for any $K \in \mathbb{Z}^+$, and let $Q \subseteq [K]$ denote the set of FL clients where v's cross-client neighbor nodes $w \in N(v) \setminus V_k$ are located. We refer to them as *neighbor clients*.

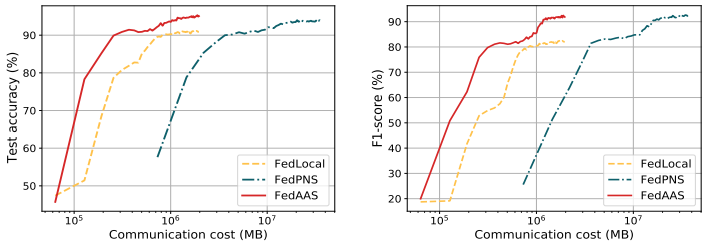
In training round t, the server randomly selects a subset M_t of m clients, and distributes the current model θ_t to them. Each chosen client k independently selects a mini-batch $B \subseteq V_k$ of nodes, fetches neighbor embeddings from both the within-client and cross-client neighbor nodes, and aggregates them to update the current node embedding using Eq. (3) in each local iteration. Then, it computes a local update $\theta_{t+1}^k \leftarrow \theta_t - \eta \frac{1}{|B|}\sum_{v\in B}\nabla f(h_v^{(L)})$ with learning rate η, and sends θ_{t+1}^k to the server. The server aggregates updates from selected clients and applies the update $\theta_{t+1} \leftarrow \frac{1}{m}\sum_{k\in M_t}\theta_{t+1}^k$. This process is iterated until the global model converges (*i.e.*, a convergence criterion is met), whence the global model $\hat{\theta}$ is obtained.

This process, however, incurs substantial computation and communication costs due to the large receptive field size and high number of cross-client neighbor nodes for large-scale graph datasets. Note that FedGCN models aggregate embeddings for each node from its neighbors in the previous layer. Then, if we track back multiple FedGCN layers, the size of supporting neighbors will grow exponentially with the depth. Specifically, evaluating all the embedding $h_v^{(l)}$ terms requires $h_v^{(l)}$ to be computed and transmitted recursively, *i.e.*, we again need the activations $h_w^{(l-1)}$ of all of v's neighbors $w \in N(v)$. For cross-client neighbor nodes, client k will send a request to the corresponding client $q \in Q$ through the server to continue neighbor sampling, and then perform cross-client embedding calculation and transmission. Suppose the average degree in a local graph G_k is d_k. To evaluate the term $h_v^{(l)}$ for one node $v \in V_k$ in an L-layer FedGCN, on average the number of neighbour nodes involved would be d_k^L [3], which leads to an exponential increase in computation and communication overhead with respect to L (*i.e.*, with $O(|\cup_{v\in V_k} N(v)\cup\{v\}|\cdot d_k^L)$ computation operations and $O(\sum_{v\in V_k}\sum_{w\in N_k\setminus V_k}\sum_{l=1}^{L} d_l d_k^L)$ communication cost per local epoch). Aggregating only within-client neighbor node embeddings and ignoring cross-client information will suffer irreducible performance loss [1, 45]. A desirable FedGCN framework should enable all participants to efficiently train a high-performance global graph model by using a small number of embedding calculations and transfers on important neighbor nodes. We assume that all participants including the server are semi-honest (*i.e.*, they follow the protocol of FL and neighbor node sampling but may be curious about others' local feature data).

4 HISTORICAL EMBEDDING-BASED ESTIMATOR

4.1 Historical Embedding for FedGCN

While computing the aggregated neighbor embedding, it is prohibitively costly to evaluate all $h_v^{(l)}$ terms because they need to be computed recursively. Our key design idea to address this challenge is to maintain a history $\bar{h}_v^{(l)}$ for each $h_v^{(l)}$ as an affordable approximation, while saving computation



(a) Test accuracy (b) F1-score

Fig. 1. Test accuracy and F1-score of the FedGCN model trained on CS dataset.

and communication overhead.

$$\tilde{h}_v^{(l+1)} \approx \gamma_\theta^{(l+1)}\Big(h_v^{(l)}, \{h_w^{(l)}\}_{w\in N(v)\cap V_k} \cup \underbrace{\{\bar{h}_w^{(l)}\}_{w\in N(v)\setminus V_k}}_{\text{Historical embeddings}}\Big). \tag{4}$$

For cross-client neighbor nodes $w \in N(v) \setminus V_k$, we approximate their embeddings $h_w^{(l)}$ via historical embeddings acquired in previous iterations, denoted as $\bar{h}_w^{(l)}$. Each client maintains a storage for all the historical embeddings. After the current local training epoch, the newly computed embeddings $h_v^{(l)}$ are pushed to the storage and serve as historical embeddings $\bar{h}_w^{(l)}$ for future epochs. The separation of within-client nodes and cross-client nodes, and their approximations via historical embeddings, significantly reduce overhead as historical embeddings are retrieved from an offline storage rather than recomputed during each iteration. Compared to the previous approach which incurs exponentially higher computation and communication cost as the number of layers L grows, that incurred by the proposed historical embedding-based estimator increases linearly with L (*i.e.*, with $O(|\cup_{v\in V_k} N(v) \cup \{v\}| \cdot L)$ computation operations and $O(\sum_{v\in V_k}\sum_{w\in N_k\setminus V_k}\sum_{l=1}^{L} d_l)$ communication cost per local epoch).

In an intuitive solution, at each local epoch, each client needs to compute the embedding of each node using all its neighbor nodes and transmit it to the target clients to serve as historical embeddings (referred to as *full synchronization*). Let τ denote the communication period, and $\tau = 1$ for full synchronization. It requires $O(\sum_{k=1}^{K} V_k| \cup_{v\in V_k} N(v) \cup \{v\}| \cdot LTJ)$ computation operations, and incurs $O(\sum_{k=1}^{K}\sum_{v\in V_k}\sum_{w\in N_v\setminus V_k}\sum_{l=1}^{L} d^l \cdot TJ)$ communication cost. T and J are numbers of global training rounds and local training epochs, respectively. Considering large-scale graphs with a large number of cross-client neighbor nodes, directly performing full synchronization incurs high computation and communication overhead in FedGCN systems. To reduce the communication cost as well as speed up FedGCN training, we design our method by the following two approaches.

(1) Reducing Unnecessary Neighbor Node Embedding Transmission through Importance Evaluation. Since not all neighbor nodes are equally important for node embedding aggregation, FedAAS leverages attention-based mechanism to iteratively quantify neighbor node influence and update their selection probabilities for node aggregation. In this way, only important cross-client neighbor node embeddings are transmitted to the corresponding FL clients, thereby avoiding unnecessary expensive communication for unimportant ones.

(2) Reducing Communication Frequency through Adaptive Historical Embedding Synchronization. A trade-off between convergence speed and prediction error can be observed when historical embedding synchronization intervals are varied. On the one hand, a longer interval saves communication delay and achieves faster convergence with respect to the wall-clock time due to lower time cost per epoch and less frequent historical embedding synchronization. On the other hand, a longer interval leads to inferior error-convergence models since the staleness of

historical embeddings results in larger embedding approximation errors. Thus, we propose an adaptive historical embedding synchronization strategy. With such a strategy, we aim to improve the error-runtime trade-off by achieving faster convergence with lower computation and communication costs as well as lower prediction errors as illustrated in Fig. 1. Here, FedLocal is the federated GraghSAGE [16], which performs random selection of within-client neighbor nodes, where the cross-client neighbor node information is ignored. FedPNS performs periodic selection and embedding synchronization of cross-client neighbor nodes.

4.2 Analysis of Approximation Error

The advantages of utilizing historical embeddings $\bar{h}_v^{(l)}$ to compute an approximation $\tilde{h}_v^{(l)}$ of the exact embedding $h_v^{(l)}$ come at the cost of an approximation error $||\tilde{h}_v^{(l)} - h_v^{(l)}||$. It can be decomposed into two sources of variances: 1) the closeness of estimated inputs to their exact values (*i.e.*, $||\tilde{h}_v^{(l-1)} - h_v^{(l-1)}||$); and 2) the staleness of historical embeddings (*i.e.*, $||\bar{h}_v^{(l-1)} - \tilde{h}_v^{(l-1)}||$). Here, we show concrete bounds for these two types of errors. Our analysis focuses on arbitrary $\gamma_\theta^{(l)}$ GCN layers as described in Eq. (1). We restrict both $A_\theta^{(l)}$ and $U_\theta^{(l)}$ to model α-Lipschitz continuous functions due to their potentially highly non-linear nature. We use the expected gradient norm as an indicator of convergence [14] since the objective function is non-convex. The algorithm produces an ϵ-suboptimal solution if

$$\mathbb{E}\Big[\min_{t\in[T]} ||\nabla F(\tilde{h}^{(L)}, \theta_t)||^2\Big] \leq \epsilon. \tag{5}$$

When ϵ is arbitrarily small, this condition guarantees algorithm convergence to a stationary point.

Assumption 1. *Let F be differentiable and λ-Lipschitz smooth, and the value of F be bounded below by a scalar F_{inf}. Let $A_\theta^{(l)}, U_\theta^{(l)}$ be Lipschitz continuous functions with Lipschitz constants α_1, α_2.*

Lemma 1. *Under Assumption 1, if for all $v \in V_k$, the inputs are close to the exact inputs (i.e., $||\tilde{h}_v^{(l-1)} - h_v^{(l-1)}|| \leq \delta$), and the historical embeddings do not become stale (i.e., $||\bar{h}_v^{(l-1)} - \tilde{h}_v^{(l-1)}|| \leq \beta$), the output error is bounded by:*

$$||\tilde{h}_v^{(l)} - h_v^{(l)}|| \leq \delta\alpha_2 + (\delta + \beta)\alpha_1\alpha_2|N(v)|. \tag{6}$$

Due to the characteristics of Lipschitz constants in the function composition, we obtain the upper bound which depends on α_1, α_2 and $|N(v)|$, as well as the errors δ and β of the inputs. We then analyze the final output error produced by a L-layer FedGCN.

Theorem 1. *Under Assumption 1, if for all $v \in V_k$ and all $l \in \{1, 2, \cdots, L-1\}$, the historical embeddings do not become stale (i.e., $||\bar{h}_v^{(l)} - \tilde{h}_v^{(l)}|| \leq \beta^{(l)}$), the final output error of layer L in round $t \in [T]$ is bounded by:*

$$||\tilde{h}_v^{(L)} - h_v^{(L)}|| \leq \sum_{l=1}^{L-1} \beta^{(l)} \alpha_1^{L-l} \alpha_2^{L-l} |N(v)|^{L-l}. \tag{7}$$

Notably, this upper bound does not depend on $||\tilde{h}_v^{(l-1)} - h_v^{(l-1)}|| \leq \delta$. Furthermore, Theorem 1 lets us immediately derive an upper error bound for the gradients, *i.e.*,

$$||\nabla f(\tilde{h}_v^{(L)}) - \nabla f(h_v^{(L)})|| \leq \lambda ||\tilde{h}_v^{(L)} - h_v^{(L)}||. \tag{8}$$

This shows that the historical embedding-based estimator incurs low variance and bias in the learning process.

5 ADAPTIVE ATTENTION-BASED SAMPLING

5.1 System Overview

FedAAS consists of two modules (as shown in Fig. 2).

(1) **Graph Attention-based Cross-client Neighbor Node Sampling.** In each embedding synchronization iteration, each selected FL client k calculates the influence scores of its neighbor nodes and sends requests to neighbor clients to conduct neighbor node sampling. Each neighbor client samples the most important cross-client neighbor nodes with the highest probabilities and sends their embeddings for historical embedding updating and node aggregation. The influence scores are updated through a shared graph attention mechanism based on the updated historical embeddings.

(2) **Adaptive Historical Embedding Synchronization and Model Updating.** With the influence estimation and sampling results, each selected client k updates its historical embeddings and performs node aggregation. Then, it updates its local model and estimates the next optimal synchronization interval via joint analysis of the overhead and error-convergence to optimize the error-runtime trade-off. Finally, client k sends the updated local model to the FL server, which then aggregates the received local models to produce the global FL model.

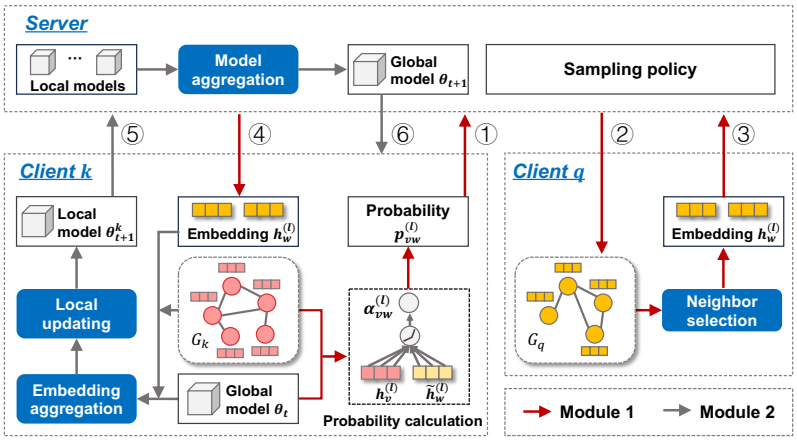


Fig. 2. System overview of FedAAS. Notations: ①-② probability $p^{(l)}_{vw}$, ③-④ node embedding $h^{(l)}_w$, ⑤ local model θ^k_{t+1}, ⑥ global model θ_{t+1}.

5.2 Graph Attention-based Neighbor Node Influence Evaluation

For efficient and accurate training, we aim to dynamically quantify the influence of neighbor nodes on the current nodes, and increase the probability of transmission and embedding aggregation for highly influential cross-client neighbor nodes. Specifically, for client $k \in [K]$, we first instantiate Eq. (1) and perform self-attention on its nodes. A shared attentional mechanism $e : \mathbb{R}^{d_l} \times \mathbb{R}^{d_l} \to \mathbb{R}$ computes a score $e(v, w)^{(l)}$ for every edge (w, v), which indicates the importance of the cross-client neighbor node w's historical representation $\bar{h}^{(l)}_w$ for client $q \in Q$ to node $v \in V_k$ in the l-th layer,

$$e(v, w)^{(l)} = \text{LeakyReLU}(a^\top \cdot [\theta^{(l)}_k h^{(l)}_v || \theta^{(l)}_q \bar{h}^{(l)}_w]) \tag{9}$$

where $a \in \mathbb{R}^{2d_l}$, $\theta^{(l)}_k \in \mathbb{R}^{d_l \times d_{l-1}}$ are learned through training. $\theta^{(l)}_k$ denotes the model parameters of the l-th FedGCN layer and $||$ is the concatenation operation. The importance values $e(v, w)^{(l)}$ for

within-client neighbor nodes $w \in N(v) \cap V_k$ can be calculated in a similar manner. These attention scores are normalized across all neighbors $w \in N(v)$ using the softmax function. Then, the probability $p_{vw}^{(l)}$ of the neighbor node w being selected to participate in the embedding aggregation of node v can be updated as,

$$p_{vw}^{(l)} = \text{softmax}(e(v,w)^{(l)}) = \frac{\exp(e(v,w)^{(l)})}{\sum_{w \in N(v)} \exp(e(v,w)^{(l)})}. \tag{10}$$

Client k then asks each client $q \in Q$ to select a set S_q of $\max\{\phi|N(v) \setminus V_k|, 1\}$ nodes from the cross-client neighbor set $N(v) \setminus V_k$ with probabilities $\{p_{v,w}^{(l)}\}$, calculate $\sum_{w \in S_q} \theta_q^{(l)} h_w^{(l)}$ and transmit them to client k through the server. Finally, client k individually selects a set of S_k of $\max\{\phi|N(v) \setminus V_k|, 1\}$ nodes from the within-client neighbor set $N(v) \setminus V_k$ and calculates the output $h_v^{(l+1)}$ by using the selected neighbor embeddings,

$$\tilde{h}_v^{(l+1)} = \sigma\Big(\sum_{w \in S_k} \theta_k^{(l)} h_w^{(l)} + \sum_{q \in Q} \sum_{w \in S_q} \theta_q^{(l)} \bar{h}_w^{(l)} \Big) \tag{11}$$

where σ is a nonlinear function. To stabilize the selection probability learning process, we can adopt the multi-head attention strategy [38] to execute multiple independent attention mechanisms and concatenate the resulting embeddings into the output embeddings.

5.3 Federated Adaptive Attention-based Sampling

Overhead Analysis for Periodic Synchronization. We now present a comparison of the overhead incurred by periodic synchronization with that incurred by full synchronization to illustrate how increasing τ achieves significant overhead reduction. In each training round t, each selected client $k \in M_t$ performs τ node embedding aggregations and local model updating before fetching embeddings from both within-client and cross-client neighbor clients. Let $C_{k,\tau}^t$ and O_τ^t be the corresponding computation time and communication delay, and $O_\tau^t b_t$ is the communication cost which is incurred by transmitting embeddings from cross-client neighbor clients with b_t being the average network bandwidth during t.

Full synchronization is equivalent to periodic synchronization with communication period $\tau = 1$. The total time to complete each iteration is $C_{\text{syn}} = \max\{C_{1,1}^t, \cdots, C_{k,1}^t\} + O_\tau^t$. In periodic synchronization, the average computation time per local updating epoch is $C_{\text{avg}} = \max\{\bar{C}_1^t, \cdots, \bar{C}_K^t\} + O_\tau^t/\tau_t$ where $\bar{C}_k^t = \frac{1}{\tau_t} \sum_{i=1}^{\tau_t} C_{k,i}^t$. We evaluate the speed-up of periodic synchronization over full synchronization to illustrate how different values of $C_{k,\tau}^t$ and O_τ^t affect the speed-up. Consider the simplest case where $C_{k,\tau}^t = C$ and $O_\tau^t = O$ are constants, and C/O is the ratio of communication delay to computation cost. The ratio depends on many factors, *e.g.*, the size of FedGCN model and the mini-batch size as well as network bandwidth, and client computing capacity. Then, the ratio of C_{syn} and C_{avg} is $C_{\text{syn}}/C_{\text{avg}} = \frac{1+C/O}{1+C/O\tau}$.

Joint Analysis of Overhead and Error-Convergence. We combine the above overhead analysis with error-convergence analysis that inspires the design of `FedAAS`.

Assumption 2. *The stochastic gradient evaluated on the mini-batch B is an unbiased estimator of the full batch gradient with bounded variance, $\mathbb{E}[g(\tilde{h}^{(L)}) - \nabla F_k(h^{(L)})] \leq \zeta^2$, where $g(\tilde{h}^{(L)}) = \frac{1}{|B|} \sum_{v \in B} \nabla f_k(\tilde{h}_v^{(L)})$.*

Theorem 2. *For periodic synchronization, under Assumption 1-2, Theorem 1 and Eq.* (8)*, if the learning rate η satisfies $\eta\lambda + \eta^2\lambda^2\tau(\tau-1) \leq 1$, C and O are local computation time and communication delay, and θ_1 is the initial model generated by the server, then after a total time cost of C_{total}, the*

minimal expected squared gradient norm is bounded by

$$\frac{2(F(\tilde{h}^{(L)},\theta_1)-F_{inf})}{\eta C_{total}}(C+\frac{O}{\tau})+\eta^2\lambda^2\zeta^2(\tau-1). \tag{12}$$

Proof. Under assumptions 1-2, after T iterations, we have:

$$\mathbb{E}\Big[\min_{t\in[T]}||\nabla F(\tilde{h}^{(L)},\theta_t)||^2\Big]\leq\mathbb{E}\Big[\frac{1}{T}\sum_{t=1}^{T}||\nabla F(\tilde{h}^{(L)},\theta_t)||^2\Big]\leq\frac{2(F(\tilde{h}^{(L)},\theta_1)-F_{inf})}{\eta T}+\eta^2\lambda^2\zeta^2(\tau-1). \tag{13}$$

Since the expected runtime per iteration for periodic synchronization is $C_{\text{avg}}=C+\frac{O}{\tau}$, the total runtime for T iterations is $C_{\text{total}}=T(C+\frac{O}{\tau})$. Directly substituting $T=C_{\text{total}}/(C+\frac{O}{\tau})$ into Eq. (13), we thus complete the proof. □

From the optimization error bound in Eq. (12), the error-runtime trade-off for different synchronization communication intervals can be derived. While a larger τ reduces the runtime per iteration and makes the first term in Eq. (12) smaller, it also adds noise and increases the last term.

The FedAAS Algorithm. Based on the joint analysis of overhead and error-convergence, we propose FedAAS to balance the first term and the last term in Eq. (12). This is achieved by starting with infrequent embedding synchronization for improved convergence speed, and gradually transiting to higher embedding synchronization frequencies to reduce the prediction error of the learned global model. The key idea of FedAAS is to select the optimal embedding synchronization interval that minimizes the optimization error at each wall-clock time. At each training round t, the server determines the optimal embedding transmission interval that achieves the fast test loss decay of the global model θ_t for the next interval. Theorem 2 shows that there is an optimal value τ^* that minimizes the optimization error bound at round t, which is given as the following.

Theorem 3. *For periodic synchronization, under assumptions 1-2, the optimization error upper bound in Theorem 2 at time C_{total} is minimized when the synchronization period is:*

$$\tau^*=\sqrt{\frac{2(F(\tilde{h}^{(L)},\theta_1)-F_{inf})O}{\eta^3\lambda^2\zeta^2C_{\text{total}}}}. \tag{14}$$

Proof. Taking first-order derivative of Eq. (12) with respect to the synchronization period and setting it to zero, we obtain:

$$\frac{2(F(\tilde{h}^{(L)},\theta_1)-F_{inf})\mathbb{E}[O]}{\eta\tau^2C_{\text{total}}}=\eta^2\lambda^2\zeta^2. \tag{15}$$

Since the term $\eta^2\lambda^2\zeta^2$ of Eq. (15) is greater than zero, the optimal synchronization period is expressed as Eq. (14). □

Suppose all clients start from the same model $\theta_1=\theta_{j=0}$ initialized by the FL server, where j denotes the wall-clock time. Applying Theorem 3 to the first time interval with $j=0$ yields the optimal embedding synchronization period as:

$$\tau_0=\sqrt{\frac{2(F(\tilde{h}^{(L)},\theta_{j=0})-F_{inf})O}{\eta^3\lambda^2\zeta^2C_{\text{total}}}}. \tag{16}$$

Algorithm 1: FedAAS

Input : K clients $\{1, 2, \cdots, K\}$, each client k owns dataset $D_k = (G_k, Y_k)$ with $G_k = (V_k, E_k)$
Output: Optimal global model $\hat{\theta}$

// At the FL Server:
Server initializes model θ_1, count $i = 0$;
for *each round* $t = 1, 2, \cdots, T$ **do**
  $M_t \leftarrow$ randomly select m clients;
  for *each client* $k \in M_t$ ***in parallel*** **do**
    $\theta^k_{t+1}, i \leftarrow$ `LocalUpdate`(k, θ_t, i)
  $\theta_{t+1} \leftarrow \frac{1}{m} \sum_{k \in M_t} \theta^k_{t+1}$ // update global model
// At FL Client $k \in [K]$:
Function `LocalUpdate`(k, θ^k_0, i):
for *each local epoch* $j = 1, 2, \cdots, J$ **do**
  Randomly select a batch $B \subseteq V_k$ of nodes, $i = i + 1$
  $Q \leftarrow$ neighbor clients with $w \in N(v) \setminus V_k$, $v \in B$
  for *each layer* $l = 1, 2, \cdots, L-1$ **do**
    $\tilde{h}^{(l)}_{w,q} = 0$ for $w \in N(v) \setminus V_k$, $q \in Q$, calculate $p^{(l)}_{v,w}$ with Eq. (10);
    $S_k \leftarrow$ select nodes $w \sim p^{(l)}_{v,w}$, $w \in N(v) \cap V_k$;
    if $l = 1$ **then**
      $\tilde{h}^{(l+1)}_v = \sigma(\sum_{w \in S_k} \theta^{k,(l)}_j h^{(l)}_w)$;
    else if $l > 1$ && $i = \tau$ **then**
      $\tilde{h}^{(l)}_{w,q} \leftarrow$ `Sampling`$(q, p_{v,w})$, $\tilde{h}^{(l+1)}_v = \sigma(\sum_{w \in S_k} \theta^{k,(l)}_j h^{(l)}_w + \sum_{q \in Q} \tilde{h}^{(l)}_{w,q})$;
      Update interval τ with Eq. (14), and $i = 0$;
    else
      $\tilde{h}^{(l+1)}_v = \sigma(\sum_{w \in S_k} \theta^{k,(l)}_j h^{(l)}_w + \sum_{q \in Q} \tilde{h}^{(l)}_{w,q})$;
  $\theta^k_j \leftarrow \theta^k_{j-1} - \eta \frac{1}{|B|} \sum_{v \in B} \nabla f(\tilde{h}^{(L)}_v, y_v)$;
Return θ^k_J, i
Function `Sampling`$(q, p_{v,w})$:
  $S_q \leftarrow$ select nodes with $w \sim p_{v,w}$, $w \in N(v) \setminus V_k$;
  $h^{(l)}_{w,q} = \sum_{w \in S_q} \theta^{q,(l)}_j h^{(l)}_w$ for local epoch $j \in [J]$;
  Return $h^{(l)}_{w,q}$;

Similarly, for the i-th time interval, clients can be viewed as restarting local training at a new initial point $\theta_{j=iT_0}$ after performing embedding synchronization. By applying Theorem 3, we have:

$$\tau_i = \sqrt{\frac{2(F(\tilde{h}^{(L)}, \theta_{j=iT_0}) - F_{inf})O}{\eta^3 \lambda^2 \zeta^2 C_{\text{total}}}}. \tag{17}$$

Interpreting Synchronization Interval. It can be observed from Eq. (16) and Eq. (17) that when the learning rate is fixed, the generated synchronization sequence decreases with increasing target value $F(\tilde{h}^{(L)}, \theta_j)$. It is consistent with the intuition that the trade-off between error-convergence

and communication efficiency varies over time. Compared to the initial phase of model training, the benefit of using a large communication period diminishes as the model approaches convergence (*i.e.*, a lower prediction error is preferred over faster runtime in latter stage of FL modeling training). Thus, FedAAS starts with infrequent embedding synchronization to improve convergence speed, and then increases the frequency to gradually reduce the variance of gradients and construct a global FedGCN model with low prediction errors.

Practical Considerations. Furthermore, in some practical scenarios where the Lipschitz constant λ and the gradient variance bound ζ^2 are unknown, estimating these constants is difficult due to highly non-convex and high-dimensional loss surfaces. To this end, we propose a simpler alternative form of FedAAS that approximates F_{inf} by 0, and divide Eq. (17) by Eq. (16) to obtain the communication period update rule $\tau_i = \left\lceil (F(\tilde{h}^{(L)}, \theta_{j=iT_0})/F(\tilde{h}^{(L)}, \theta_{j=0}))^{\frac{1}{2}} \tau_0 \right\rceil$, where $\lceil r \rceil$ is the ceil function to round r to the nearest integer. Since the objective function values (*i.e.*, test loss $F(\tilde{h}^{(L)}, \theta_{j=iT_0})$ and $F(\tilde{h}^{(L)}, \theta_{j=0})$) can be easily obtained during aggregation, we get a heuristic estimate of τ_0 by running a simple grid search over different τ run for one or two epochs each.

Implementation. The proposed FedAAS approach is illustrated in Algorithm 1. Specifically, the FL server initializes a global model θ_1 and a global count $i = 0$. In the t-th training round, the server randomly selects a subset of m clients, M_t, and distributes the current model θ_t to them. Each selected client $k \in M_t$ at local epoch j first independently selects a batch $B \subseteq V_k$ of nodes $v \in V_k$. For each layer l of FedGCN, client k calculates the importance score $\xi_{vw}^{(l)}$, sampling probability $p_{vw}^{(l)}$ and selects sets S_k, S_q of within-client and cross-client neighbors with the sampling ratio ϕ. Then, it calculates the embedding $\tilde{h}_v^{(l+1)}$ following Eq. (11) by aggregating selected historical embeddings $\tilde{h}_{w,q}^{(l)}$ of cross-client neighbors and selected embeddings $\theta_j^{k,(l)} h_w^{(l)}$ of within-client neighbors (line 25). When the number of local epochs j reaches τ (*i.e.*, $i = \tau$), client k performs embedding synchronization by asking client $q \in Q$ to select and update the selected cross-client neighbor embeddings and transmit them back to k (lines 29-32). Finally, client k sets $i = 0$, updates the synchronization interval τ (line 23) and local model parameters θ_j^k, and sends the model θ_J^k to the FL server (lines 26-27).

6 ANALYTICAL EVALUATION

6.1 Privacy Analysis

FedAAS preserves each client's local feature data from exposure to other parties, including the FL server, during the sampling and updating process. Firstly, the training process strictly follows a standard FL training protocol. Hence, no local feature data are transmitted during training [29]. Secondly, during the sampling process, no local feature data are transmitted. Let clients k and $q \in [K]$ be neighboring clients who need to share embeddings during the sampling process. Suppose k aggregates embeddings from q, and intends to infer q's original node features $H_q^{(1)} = \{h_w^{(1)} | w \in V_q\}$, which is a matrix containing features of all nodes held by q (*i.e.*, $h_w^{(1)} = x_w$). Thanks to our attention-based neighbor sampling design, q transmits only the selected aggregated embeddings $h_{w,q}^{(l)} = \sum_{w \in S_q} \theta_j^{q,(l)} h_w^{(l)}, j \in [J]$ (Algorithm 1 line 29) to k when they synchronize embeddings. Thus, it is difficult for k to infer individual $h_w^{(2)}, \cdots, h_w^{(L)}$ with aggregated embeddings $h_{w,q}^{(l)}$ although it approximates the remote model $\theta_j^{q,(l)}$ using its local model $\theta_j^{k,(l)}$ [1].

Last but not least, though $h_w^{(2)}, \cdots, h_w^{(L)}$ can be obtained somehow, it is still difficult for k to accurately infer q's node features $H_q^{(1)}$. Let $Q_1 \subseteq [K]$ denote the set of clients where w's cross-client neighbors are located, and S_q, S_k denote the sets of within-client and cross-client neighbors sampled

with the probability $p_{vw}^{(1)}$. Since $h_w^{(2)} = \sigma(\sum_{v\in S_q} \theta_q^{(1)} h_v^{(1)} + \sum_{k\in Q_1} h_{w,q}^{(1)})$, client k has no information about the selected set S_q and w's neighbor set Q_1. That is, k has no information about either $\sum_{v\in S_q} \theta_q^{(1)} h_v^{(1)}$ or $\sum_{k\in Q_1} \sum_{v\in S_v} \theta_k^{(1)} h_v^{(1)}$. Thus, it is difficult for client k to infer q's original node features $H_q^{(1)}$. Therefore, FedAAS can protect node features, while enabling information sharing during sampling and updating process.

6.2 Convergence Analysis

Here, we present a convergence guarantee for FedAAS by extending the error analysis for synchronization. Without loss of generality, we analyze an arbitrary synchronization period sequence $\{\tau_1, \cdots, \tau_R\}$ with R embedding synchronization rounds.

Theorem 4. *For* FedAAS*, suppose the learning rate remains the same in each local model updating period. If the following conditions are satisfied as* $R \to \infty$*,*

$$\sum_{r=0}^{R} \eta_r \tau_r \to \infty, \sum_{r=0}^{R} \eta_r^2 \tau_r < \infty, \sum_{r=0}^{R} \eta_r^3 \tau_r^2 < \infty, \tag{18}$$

then the global model θ *is guaranteed to converge to a stationary point:*

$$\mathbb{E}\Big[\frac{\sum_{r=0}^{R-1} \eta_r \sum_{t=1}^{\tau_r} ||\nabla F(\tilde{h}^{(L)}, \theta_{\sum_{i=0}^{r-1}\tau_i + k})||}{\sum_{r=0}^{R-1} \eta_r \tau_r}\Big] \to 0. \tag{19}$$

The basic idea of proof is as follows. First, to understand the meaning of condition (18), let us consider the case when $\tau_0 = \cdots = \tau_R$ is a constant. Then, the converge condition is identical to mini-batch SGD: $\sum_{r=0}^{R} \eta_r \to \infty, \sum_{r=0}^{R} \eta_r^2 < \infty$. As long as the sequence of communication periods is bounded, the learning rate scheme in mini-batch SGD can be easily adjusted to satisfy condition (18). In particular, when the communication period sequence decreases, the last two terms in (18) become easier to satisfy, and the differences between the objective values of two consecutive rounds are bounded. The proof details are illustrated as follows.

Proof. Let matrices $\Theta_t, g_t \in \mathbb{R}^{d\times K}$ that concatenate all local models and gradients, which are $\Theta_t = [\theta_t^1, \cdots, \theta_t^K]$ and $g_t = \{g(\tilde{h}^{(L)}, \theta_t^1), \cdots, g(\tilde{h}^{(L)}, \theta_t^K)\}$. We focus on the r-th local update period, where $r \in \{0, 1, \cdots, R\}$. Without loss of generality, suppose the index of the r-th local update period starts from 1 and ends at τ_r. Then, for the j-th local step, we have the following lemmas.

Lemma 2. *For* FedAAS*, under assumptions 1-2, at the* j*-th iteration, we have the following bound for the objective value:*

$$\mathbb{E}[F(\tilde{h}_v^{(L)}, \theta_{t+1}) - F(\tilde{h}_v^{(L)}, \theta_t)] \le -\frac{\eta_r}{2}(1 - \eta_r\lambda) \cdot \frac{||\nabla F(\Theta_t)||^2}{K} + \frac{\eta_j^2 \lambda \zeta^2}{2K} + \frac{\eta_j \lambda^2}{2K} ||\Theta_t(\mathbf{I} - \mathbf{J})||^2 \tag{20}$$

where $\mathbf{J} = \mathbf{1}\mathbf{1}^{\mathbf{T}}/(\mathbf{1}^{\mathbf{T}}\mathbf{1})$ *is a* $K \times K$ *matrix (*$\mathbf{1} = [1, \cdots, 1]^T$*) and the identity matrix* $\mathbf{I}$ *is of size* $K \times K$*.*

Taking the expectation and summing over all iterations during the r-th local update, we have:

$$\begin{aligned}\mathbb{E}[F(\tilde{h}^{(L)}, \theta_{\tau_r+1}) - F(\tilde{h}^{(L)}, \theta_1)] \le &-\frac{\eta_r}{2}\sum_{t=1}^{\tau_r} \mathbb{E}||\nabla F(\tilde{h}^{(L)}, \theta_t)||^2 - \frac{\eta_r}{2}(1 - \eta_r\lambda) \cdot \sum_{t=1}^{\tau_r} \frac{\mathbb{E}[||\nabla F(\Theta_t)||^2]}{K} \\ &+ \frac{\eta_r^2 \lambda \zeta^2 \tau_r}{2K} + \frac{\eta_r \lambda^2}{2K} \sum_{t=1}^{\tau_r} \mathbb{E}||\Theta_t(\mathbf{I} - \mathbf{J})||^2.\end{aligned} \tag{21}$$

Note that $\Theta_t(\mathbf{I}-\mathbf{J}) = -\eta_r \sum_{i=1}^{t-1} g_i(\mathbf{I}-\mathbf{J})$, and it follows the fact that all clients start from the same model in the beginning (*i.e.*, $\Theta_1(\mathbf{I}-\mathbf{J}) = 0$). Thus, we have:

$$\mathbb{E} \leq \underbrace{2\eta_r^2 \sum_{k=1}^{K} \mathbb{E}[|| \sum_{i=1}^{t-1} g(\tilde{h}^{(L)}, \theta_i^k) - \nabla F_k(\tilde{h}^{(L)}, \theta_i^k)||^2]}_{H_1} + \underbrace{2\eta_r^2 \sum_{k=1}^{K} \mathbb{E}[|| \sum_{i=1}^{t-1} \nabla F_k(\tilde{h}^{(L)}, \theta_i^k)||^2]}_{H_2}. \tag{22}$$

For the first term H_1, since the stochastic gradients are unbiased, all cross terms are zero. Thus, combining with Assumption 2, we have $H_1 \leq 2\eta_r^2 K(t-1)\zeta^2$. For the second term H_2, directly applying Jensen's inequality yields:

$$H_2 \leq 2\eta_r^2 (t-1) \sum_{i=1}^{t-1} \mathbb{E}[||\nabla F(\Theta_i)||^2]. \tag{23}$$

Substituting the bounds of H_1 and H_2 into Eq. (22), we have:

$$\mathbb{E}[||\Theta_t(\mathbf{I}-\mathbf{J})||^2] \leq 2\eta_r^2 (t-1) \sum_{i=1}^{t-1} \mathbb{E}||\nabla F(\Theta_i)||^2 + 2\eta_r^2 \zeta^2 K(t-1). \tag{24}$$

Recall the upper bound Eq. (20), we derive the following bound:

$$\sum_{t=1}^{\tau_r} \mathbb{E}||\Theta_t(\mathbf{I}-\mathbf{J})||^2 \leq \sum_{t=1}^{\tau_r} \eta_r^2(\tau_r^2 + \tau_r)(t-1) \sum_{i=1}^{t-1} \mathbb{E}||\nabla F(\Theta_i)||^2 + \eta_r^2 K \zeta^2 \tau_r(\tau_r - 1). \tag{25}$$

Substituting Eq. (25) into Eq. (20), and when the learning rate satisfies $\eta_r^2\lambda^2\tau_r(\tau_r-1) + \eta_r\lambda \leq 1$,

$$\mathbb{E}[F(\tilde{h}_v^{(L)}, \theta_{\tau_r+1}) - F(\tilde{h}_v^{(L)}, \theta_1)] \leq -\frac{\eta_r}{2} \sum_{t=1}^{\tau_r} \mathbb{E}||\nabla F(\theta_i)||^2 + \frac{\eta_r^2\lambda\zeta^2\tau_r}{2K} + \frac{\eta_r^3\lambda^2\zeta^2\tau_r(\tau_r-1)}{2}. \tag{26}$$

Suppose $s_r = \sum_{i=0}^{r-1} \tau_i + 1$ is the first index in the r-th local period. Summing over all local periods from 0 to R, we obtain:

$$\mathbb{E}\Big[\frac{\sum_{r=0}^{R-1} \eta_r \sum_{t=1}^{\tau_r} ||\nabla F(\tilde{h}^{(L)}, \theta_{s_i+k})||}{\sum_{r=0}^{R-1} \eta_r\tau_r}\Big] \leq \frac{\lambda\zeta^2 \sum_{r=0}^{R-1} \eta_r^2\tau_r}{K \sum_{r=0}^{R-1} \eta_r\tau_r} + \frac{2(F(\tilde{h}^{(L)}, \theta_1) - F_{inf})}{\sum_{r=0}^{R-1} \eta_r\tau_r} + \lambda^2\zeta^2 \frac{\sum_{r=0}^{R-1} \eta_r^3\tau_r}{\sum_{r=0}^{R-1} \eta_r\tau_r}. \tag{27}$$

When the condition Eq. (18) holds as $R \to \infty$, the bound Eq. (27) converges to zero. Thus, we complete the proof of convergence. □

7 EXPERIMENTAL EVALUATION

7.1 Experimental Settings

7.1.1 Implementation. We have implemented FedAAS and in an FL system consisting of one server and 50 clients. To further investigate the performance of FedAAS in large-scale FL systems, we also tested it in an environment with up to 150 clients. Note that there are two main modules in our implementation, 1) federated graph learning framework with cross-client embedding transmission and aggregation, and 2) adaptive historical embedding synchronization. The implementation of the first module is independent of our algorithm design since all existing related work [1, 7] on federated graph learning needs to implement such a framework. We leverage the widely adopted PyTorch Geometric library that uses the message passing mechanism to build graph neural networks (GNNs). Then, we extend GNNs into FL settings by adopting existing FL frameworks [29]. For the second module, we update the embedding interval and the aggregated embeddings by using the parameters

generated during the first module. Our implementation is based on Python 3.11, Pytorch 2.1 and Pytorch Geometric 2.0.1 [12]. All the experiments are performed on Ubuntu 20.04 operating system equipped with a 32-core AMD Ryzen Threadripper PRO 5965WX s (48) @ 3.800GHz CPU, 192G of RAM and three NVIDIA RTX A5000 GPUs, each having 24GB memory.

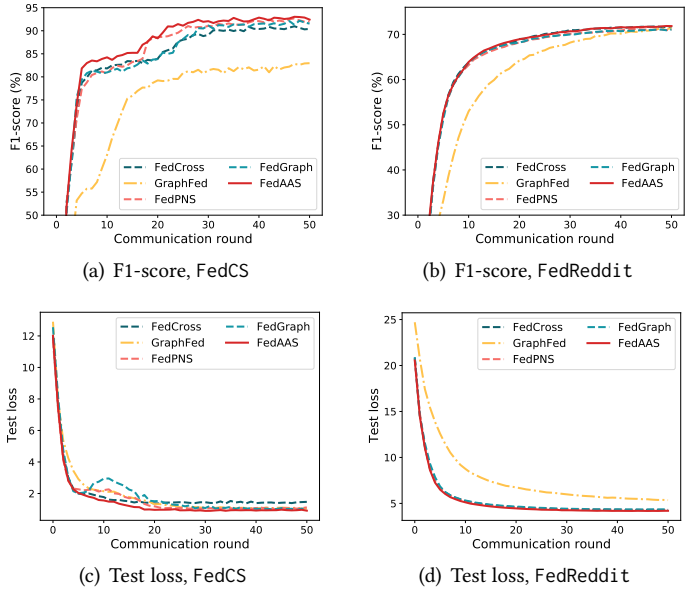
7.1.2 Datasets. We use six real-world graph datasets of different scales for our FedGCN tasks. **1) CS** and **2) Physics** [35]: These two datasets are co-authorship graphs based on the Microsoft Academic Graph. Here, nodes represent authors that are connected by an edge if they coauthored a paper; node features represent paper keywords for each author's papers. **3) Pubmed** [34]: This dataset consists of 19,717 scientific publications pertaining to diabetes where publication is described by a TF/IDF weighted word vector. **4) Amazon Computers (AC)**: [34] Nodes represent goods and edges represent that two goods are frequently bought together. Given product reviews as bag-of-words node features, the task is to map goods to their respective product category. **5) Yelp** [43]: This dataset is a subset of Yelp's businesses, reviews, and user data for use in connection with academic research. It was originally put together for the Yelp Dataset Challenge which is a chance for students to conduct research or analysis on Yelp's data and share their discoveries. **6) Reddit** [16]: This dataset contains Reddit posts belonging to different communities, where posts are connected if the same user comments on them.

For these datasets, we leverage their original training/ validation/ testing split ratios [12] in our evaluations. We partitioned training sets and validation sets over 50 clients in both independent and identically distributed (iid) setting and non-independent and identically distributed (non-iid) setting [29]. Take Reddit for example, we divided nodes of each class into 50 shards, and assign each client 41 shards. When the 41 shards contain nodes of 41 different categories, this is referred to as the iid setting; otherwise, non-iid one. We simulate a non-iid partition by simulating $p_i \sim Dir_k(\alpha)$ with $\alpha = 0.5$ through a Dirichlet distribution, and allocating a $p_{i,k}$ proportion of the instances of class i to client k, following the setting in [26, 42]. With $\alpha \to \infty$, all clients are iid, while with $\alpha \to 0$, each client is much non-iid, *e.g.*, each one holds examples from only one class. We partitioned other datasets in a similar way. Since the original graph is extremely dense, we downsample the edges in local subgraphs, *i.e.*, we randomly cut off 99% of the edges, following the setting in [16]. The test dataset is located at the FL server. The statistics of the datasets are presented in Table 1.

Table 1. Statistics of the datasets and the synthesized distributed subgraph system. V_k, E_k denote the averaged numbers of nodes and edges in all subgraphs of client k, $\triangle E$ denotes the total number of cross-client edges.

Dataset	CS	Physics	Pubmed	AC	Yelp	Reddit
V	18,333	34,493	19,717	13,752	716,847	232,965
E	163,788	247,962	88,648	491,722	13,954,819	114,615,892
# features	6,805	8,415	500	767	300	602
# classes	15	5	3	10	100	41
Train/Val/Test	0.8/0.1/0.1	0.8/0.1/0.1	0.8/0.1/0.1	0.8/0.1/0.1	0.75/0.10/0.15	0.66/0.10/0.24
50 clients						
V_k	293	552	364	219	10,741	3,068
E_k	32	48	18	98	2,761	22,718
$\triangle E$	2,020	6,068	1,480	6,114	144,977	1,024,925

7.1.3 FL Models. We implemented `FedAAS` (Algorithm 1) by leveraging the widely used GraphSAGE model [16] in an FL system achieving following tasks: 1) `FedCS` and 2) `FedPhysics`: mapping authors to their respective field of study given paper keywords for each author's papers; 3) `FedPubmed`: classifying scientific publication categories with the citation network; 4) `FedAC`: detecting the category of goods given product reviews; 5) `FedYelp`: predicting communities of online posts based on user comments; and 6) `FedReddit`: categorizing types of businesses based on customer



(a) F1-score, FedCS (b) F1-score, FedReddit

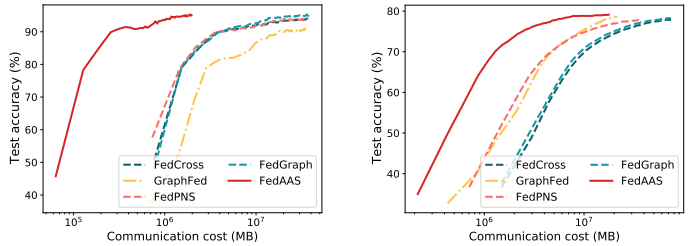
(c) Test loss, FedCS (d) Test loss, FedReddit

Fig. 3. F1-score and test loss of different communication rounds for training various FedGCN models.

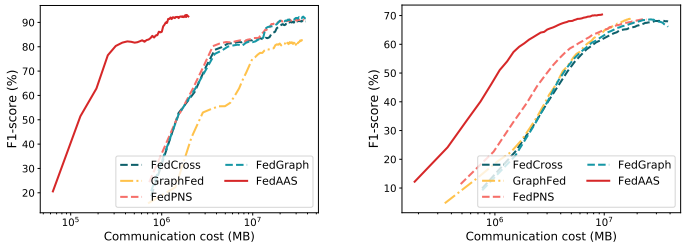
reviewers and friendship. Each model has two hidden graph attention layers with 256 and 128 neurons respectively and uses the mean aggregator. We set the number of neighbor nodes sampled in each layer to 10 and the minimum interval to 2 batch training epochs. During model training, we use Adam [22] as the optimizer with the weight decay to be zero. We leverage ReLU as the activation function and use the cross-entropy loss function. We set the initial learning rate $\eta = 0.001$, the fixed batch number is 10 and local epoch is 1. We conduct FedGCN learning until a pre-specified test accuracy is reached, or a maximum number of iterations has elapsed (*e.g.*, 50 rounds). For all experiments, we perform 10-fold cross validation and report the average results.

7.1.4 Comparison Baselines. We compare FedAAS against the following four baselines. **1) FedCross**: It performs the random neighbor node selection of both local subgraph neighbors and cross-client neighbors in each local training epoch. **2) GraphFed** [6]: It samples cross-client neighbor nodes of overlapping nodes and adopts personalized subgraph FL with graph data distribution similarity obtained using inter-subgraph distances. **3) FedPNS** [7]: It conducts the periodic neighbor node selection and embedding synchronization of cross-client neighbors. We set the periodic interval to 2 local epochs, *i.e.*, we sample cross-device neighbors every 2 local epochs, and sampling neighbor nodes of local subgraphs for the remaining epochs. **4) FedGraph** [1]: It samples neighbor nodes of local subgraph neighbors and cross-client neighbors by automatically adjusting sampling policies based on deep reinforcement learning. For all the methods, we set the neighbor node sampling size to 10, and for the method FedCross, we set both the number of local sampling nodes and the number of cross-device sampling nodes to be 5 batch training epochs.

7.1.5 Evaluation Metrics. To measure the accuracy of FedGCN models, we use three metrics [11, 19], *i.e.*, test accuracy, F1-score, and Area Under the Curve (AUC). There are four types of predictions for a model: True Positive (TP), where the model correctly predicts the positive class; True Negative (TN), where the model correctly predicts the negative class; False Positive (FP), where the model incorrectly predicts the positive class; and False Negative (FN), where the model incorrectly predicts

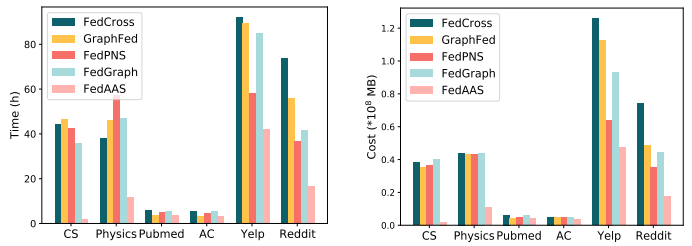


(a) Test accuracy, FedCS, iid (b) Test accuracy, FedReddit, iid



(c) F1-score, FedCS, iid (d) F1-score, FedReddit, non-iid

Fig. 4. Test accuracy and F1-score of size of communication cost for training FedGCN models.



(a) Computation cost (b) Communication cost

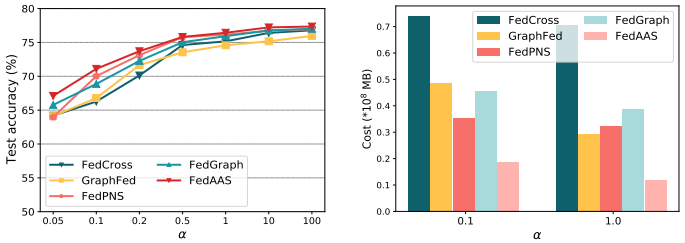
Fig. 5. The total computation cost and communication cost for training different FedGCN models.

the negative class. Based on these four possible prediction results, the performance metrics test accuracy, F1-score and AUC are defined as follows.

- **Test accuracy.** Test accuracy is a metric that measures how often a DL model correctly predicts the outcome. It can be calculated as $Test\ accuracy = \frac{TP+TN}{TP+TN+FP+FN}$.
- **F1-score.** F1-score is the harmonic mean of the precision and recall. It ranges from 0 to 1, and a higher F1-score indicates more accurate predictions. F1-score is calculated as $F1 - score = \frac{2 \cdot precision \cdot recall}{precision+recall}$, where precision is calculated as $precision = \frac{TP}{TP+FP}$ and recall (True Positive Rate, TPR) is defined as $recall = \frac{TP}{TP+FN}$.
- **AUC.** AUC is the area under the Receiver Operating Characteristic (ROC) curve that plots TPR against False Positive Rate (FPR). It ranges from 0 to 1, where 0.5 means a random prediction, and 1 means a perfect prediction.

Table 2. Performance comparison for training different FedGCN models on various datasets.

Method	Metric	Performance results (%) ±standard deviations											
		CS		Physics		Pubmed		AC		Yelp		Reddit	
		iid	non-iid	iid	non-iid	iid	non-iid	iid	non-iid	iid	non-iid	iid	non-iid
FedCross	testAcc	93.08±0.78	90.31±1.02	95.73±0.48	94.89±0.24	86.14±2.16	86.24±1.03	88.94±0.42	87.97±0.63	85.36±3.29	86.41±2.08	78.98±0.52	75.62±0.62
	F1-score	89.51±1.25	**80.68±0.94**	93.54±0.37	92.19±1.02	85.62±0.87	84.44±0.91	**88.34±0.25**	**85.17±0.81**	31.63±2.14	19.41±1.03	71.08 ±1.52	65.77±1.15
	AUC	97.89±1.32	82.23±1.28	99.21±0.39	**96.21±0.68**	95.83±0.64	95.29±0.71	98.36±0.29	91.42±0.36	73.16±2.03	77.36±1.43	95.76±0.05	81.12±1.02
GraphFed	testAcc	92.61±0.62	89.17±1.08	95.29±0.78	93.91±0.64	87.52±0.77	85.81±0.77	88.29±1.09	87.09±0.58	68.39±0.52	67.67±0.62	78.76±0.32	74.46±0.53
	F1-score	83.77±1.05	77.03±1.03	93.15±0.57	82.19±0.76	85.84±0.87	84.21±0.51	83.37±0.24	82.71±1.12	25.53 ±0.52	20.52±1.15	71.02±0.23	65.56±0.38
	AUC	97.61±0.67	80.26±0.82	98.88±0.34	95.81±0.64	95.88±0.14	94.43±0.74	98.16±1.03	90.36±1.13	72.47±0.35	70.41±1.02	95.85±0.43	79.89±0.29
FedPNS	testAcc	94.24±0.21	90.03±0.57	95.76±0.34	94.71±0.79	86.96±1.23	85.27±1.05	88.64±0.55	86.58±0.74	88.76±0.59	87.57±0.82	78.95±1.26	75.78±0.73
	F1-score	91.42±2.34	79.41±1.02	93.61±0.76	91.63±0.57	85.68±2.74	**85.81±1.46**	86.46±0.73	81.56±1.06	31.78±1.02	15.89±1.14	70.34±0.62	65.74±0.62
	AUC	98.31±2.37	78.89±1.72	99.11±0.34	94.87±1.24	95.76±1.29	94.96±1.09	98.37±0.79	89.26±0.72	73.97±0.25	76.74±1.02	**96.43±1.28**	81.07±0.36
FedGraph	testAcc	94.81±1.16	90.17±1.15	94.93±0.43	94.45±0.52	87.36±0.97	86.14±0.84	89.72±0.63	86.57±0.71	89.05±0.82	87.43±0.58	78.34±0.75	75.93±1.04
	F1-score	91.88±1.24	79.03±1.16	92.52±0.81	91.46±0.41	85.58±1.04	84.58±1.41	85.21±0.63	82.25±0.38	**31.94±1.22**	20.42±1.02	70.85±0.83	66.42±1.04
	AUC	98.58±0.78	80.86±0.76	98.87±0.61	95.13±0.17	95.81±1.62	95.75±1.27	**98.41±0.39**	85.94±0.39	74.15±0.51	78.31±0.89	96.13±1.17	79.82±0.71
FedAAS	testAcc	**95.32±0.16**	**91.62±0.75**	**95.98±0.41**	**95.02±0.28**	**88.82±0.59**	**86.51±1.07**	**89.83±0.28**	**88.13±0.68**	**90.48±0.17**	**88.46±0.79**	**79.26±0.18**	**76.12±0.83**
	F1-score	**92.91±1.07**	79.67±1.07	**93.89±0.26**	**92.65±0.37**	**86.19±0.21**	84.91±0.83	86.53±0.29	84.83±0.37	31.65±0.26	**25.87±0.69**	**71.64±0.23**	**66.38±0.51**
	AUC	**98.72±0.12**	**83.71±1.04**	**99.23±0.18**	95.12±0.31	**95.97±0.31**	**95.81±0.94**	97.82±0.45	**91.64±0.27**	**74.32±0.12**	**78.53±0.72**	96.26±1.05	**81.48±0.56**



(a) Test accuracy vs. non-iid degree (b) Communication cost

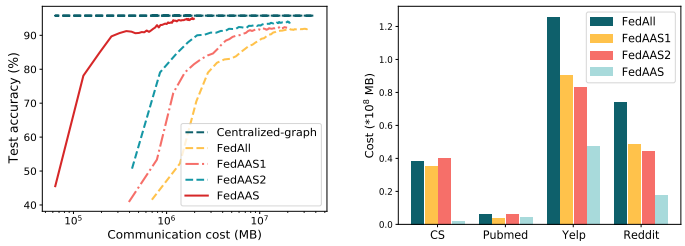
Fig. 6. The performance of model FedReddit trained on dataset Reddit with different non-iid degree settings.

7.2 Results and Discussions

FedAAS **improves accuracy.** We compare FedAAS with other baselines by training different FedGCN models, and evaluating the performance metric scores of the global models. We present the F1-score and test loss of different communication rounds of models FedCS and FedReddit for training datasets CS and Reddit in the iid setting in Fig. 3. The other metric scores of different communication rounds of models FedPubmed and FedYelp in both iid and non-iid settings are quite similar to that in Fig. 3. The results show that FedAAS outperforms all other methods in terms of both accuracy scores and convergence speed. For example, in the 50-th round of training the FedCS model on dataset CS, the F1-score of FedAAS is 92.91% outperforming other baselines by 3.4%, 9.14%, 1.49% and 1.03%. Besides, compared to FedLocal that ignores the cross-client information (see Fig. 1), all these methods that consider cross-client information can achieve much higher accuracy scores.

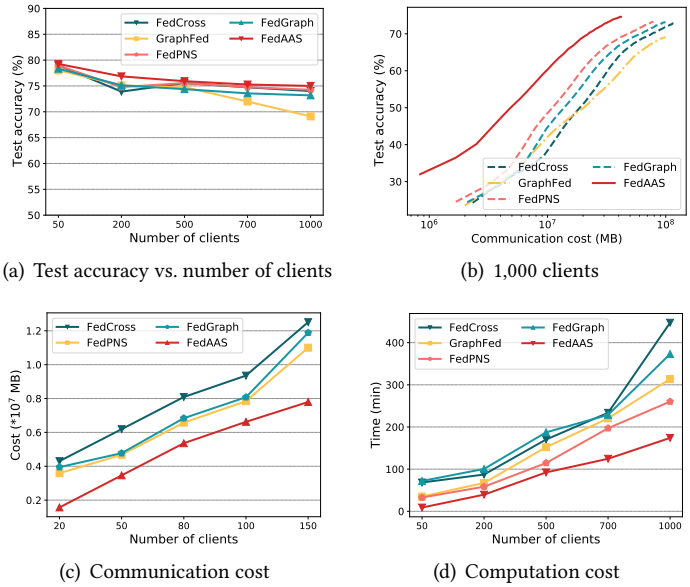
We present the results of test accuracy, F1-score, AUC scores and the standard deviations of the final global models in Table 2. It shows that FedAAS achieves higher test accuracy, F1-score and AUC scores than all other baseline methods in almost all cases. As an example, for model FedPubmed, the average test accuracy of FedAAS is 1.30% higher than the best performing baseline. Meanwhile, the standard deviations are relative small, *e.g.*, with 1.57%, 0.18%, 0.64% and 0.38% smaller than that of other baselines. For some cases where FedAAS performs second best in terms of accuracy metrics, our test accuracy, *e.g.*, 96.26% is much close to the best accuracy 96.43%, while FedAAS saves a large amount of communication and computation overhead which will be illustrated later.

FedAAS **improves efficiency.** To clearly illustrate that FedAAS achieves faster convergence with lower computation and communication costs, we present the test accuracy and F1-scores with the size of communication overhead for training models FedCS and FedReddit in iid settings in Fig. 4. The test accuracy and F1-scores with the size of communication overhead for training FedPubmed and FedYelp in both iid and non-iid settings is much similar to that in Fig. 4. Besides, the test



(a) Test accuracy (b) F1-score vs. size of communication

Fig. 7. Test accuracy and communication cost of model FedCS trained with different ablation baselines.



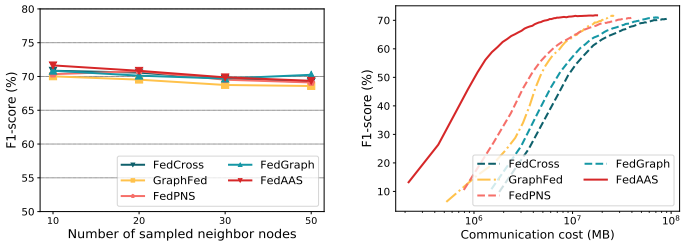
(a) Test accuracy vs. number of clients (b) 1,000 clients

(c) Communication cost (d) Computation cost

Fig. 8. The performance of model FedReddit (a-b) and FedPubmed (c-d) with different number of participants.

accuracy and F1-scores with the total runtime for training various FedGCN models is also much similar to that in Fig. 4. The results show that FedAAS requires less amount of communication volume than the baselines to achieve the target accuracy scores, which leads to less training time. For example, for the target test accuracy 80.0% of the trained FedCS, FedAAS achieves 23.83 ×, 25.19×, 22.9×, 19.34× faster runtime and its communication cost is 94.88%, 94.46%, 94.62%, 95.11% lower than the other four baselines. In addition, compared to the full synchronization methods, *i.e.*, FedCross, GraphFed, FedGraph, the periodic synchronization methods, *i.e.*, FedAAS, FedPNS achieves faster convergence with lower computation and communication costs.

Furthermore, we present the total computation cost and communication cost when training FedGCN models using different methods in Fig. 5. It can be observed that FedAAS achieves significantly saving of both computation costs and communication costs than others. As an example, for model FedReddit trained on Reddit, FedAAS achieves 22.89× faster runtime and requires less than 76.19% communication cost than the best performing baseline. Overall, FedAAS converges faster with lower computation costs and communication costs as well as lower prediction errors.



(a) F1-score (b) F1-score vs. size of communication

Fig. 9. F1-score of model FedReddit trained on dataset Reddit with different sampled neighbor nodes.

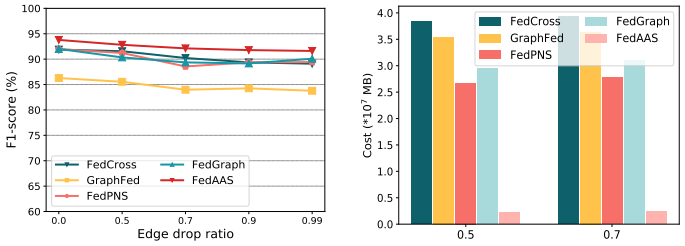
7.3 Ablation Study

We perform various ablation studies to show the effectiveness of each component of FedAAS. We compare FedAAS against the following baselines: 1) Centralized-graph: it conducts the centralized graph learning using random selected neighbors; 2) FedAll: it conducts FL training using all neighbor nodes for each node in each local training epoch; 3) FedAAS1: it only conducts the proposed attention-based neighbor node sampling without the adaptive embedding synchronization; and 4) FedAAS2: it conducts training using all neighbor nodes for each node with the proposed adaptive embedding synchronization. We train various FedGCN models using these ablation baselines and evaluating the performance of the global models. We present the test accuracy with the size of communication cost for training FedCS in Fig. 7(a) and show the total communication costs in Fig. 7(b). The results show that the test accuracy of Centralized-graph is relative high, *e.g.*, with 3.50%, 3.39%, 1.92%, 0.68% higher than that of FL baselines. Besides, FedAAS, FedAAS1 and FedAAS2 achieve higher performance in saving much communication costs to reach the target accuracy scores and FedAAS performs the best among them. Thus, both the attention-based neighbor node sampling module and the adaptive embedding synchronization module are effective to construct FedAAS.

7.4 Sensitivity Analysis

Impact of the non-iid degree. We evaluate how FedAAS behaves as the non-iid degree α varies, where α indicates the concentration parameter of the Dirichlet distribution. We present the example results of test accuracy of the model FedReddit with different non-iid degrees (*i.e.*, $\alpha = 0.05, 0.1, 0.2, 0.5, 1.0, 10, 100$) in Fig. 6(a). It can be observed that FedAAS achieves the highest test accuracy in almost all cases. Besides, these accuracy scores of the model FedReddit increase as α increases, and when α is larger than 0.5, the accuracy scores are relatively high. We further present the F1-scores with the size of communication cost for training the model FedReddit in Fig. 4(d) with $\alpha = 0.5$, and that of other non-iid degree settings are quite similar to Fig. 4(d). The results in Fig. 4(d) show that FedAAS consistently requires less communication cost than others to achieve the target test accuracy. In addition, we present two examples of communication costs under $\alpha = 0.1, \alpha = 1$ settings in Fig. 6(b), and the computation costs are quite similar to Fig. 6(b). It shows that FedAAS significantly saves both communication and computation costs than other baselines, and is highly robust against different non-iid degree settings.

Impact of the number of participants. To evaluate the scalability of FedAAS, we conduct large-scale client engagement experiments with different number of clients, *i.e.*, $K = 50, 200, 500, 700, 1,000$, for training different FedGCN models. We present the example results of test accuracy of FedReddit trained with different number of clients in Fig. 8(a). The results show that the test accuracy of FedAAS is consistently high, *e.g.*, above 75.0%, as the number of client increases to 1,000 and achieves test accuracy that is comparable to or higher than others. Besides, we present the test accuracy with



(a) F1-score (b) Communication cost

Fig. 10. F1-score and communication cost of model FedCS trained with different subgraph edge drop ratios.

the size of communication cost in Fig. 8(b) for training FedReddit with $K = 1,000$. It shows that FedAAS requires less amount of communication overhead than the baselines to achieve the target test accuracy scores. Further, we show the computation and communication costs of FedReddit in Fig. 8(c), 8(d). It shows that both communication and computation costs increase as the number of client increases and meanwhile, FedAAS achieves substantial cost savings than other baselines in all settings. For example, in the 1,000 client setting, FedAAS achieves 29.84% lower runtime and 33.11% lower communication cost than the best performing baseline. Thus, under different number of client settings, FedAAS consistently achieves the target accuracy most efficiently.

Impact of the number of neighbor nodes sampled. We investigate how FedAAS behaves as the number of sampled neighbor nodes varies. Fig. 9(a) presents the F1-scores of the trained FedReddit model when the number of sampled neighbor nodes is 10, 20, 30, 50, and the test accuracy and AUC scores are much similar to Fig. 9(a). It shows that by sampling only 10 influential neighbor nodes, FedAAS can construct a FedGCN model with high accuracy. We further present the F1-score with the size of communication cost in Fig. 9(b). It shows that FedAAS achieves much communication savings and highly efficient model training.

Impact of the local subgraph edge drop ratio. Fig. 10(a) presents the F1-score of the trained model FedCS when the subgraph edge drop ratio is 0, 0.5, 0.7, 0.9, 0.99. It shows that as the drop ratio increases, the F1-score decreases slightly, and FedAAS achieves consistently high F1-scores, *e.g.*, with the F1-score above 90%, with different drop ratios. We also present two examples of communication cost in Fig. 10(b) for ratio settings of 0.5, 0.7, and the computation costs are quite similar to Fig. 10(b). It shows that FedAAS significantly saves both communication and computation overhead than other baselines, which demonstrates the robustness of FedAAS against different subgraph edge drop ratio settings.

8 CONCLUSIONS

In this work, we proposed a federated adaptive attention-based graph sampling approach, FedAAS, for large-scale graph data in node classification tasks. It achieves substantial communication and computation cost savings by efficiently leveraging historical embedding estimators to accurately fill in the inter-dependency information of cross-client neighbors. FedAAS achieves highly efficient federated graph learning by prioritizing highly influential neighbors and adaptive embedding transmission based on historical embeddings. It reduces unnecessary cross-client neighbor embedding communications through the adaptive attention-based sampling to determine the optimal communication period and achieves faster convergence with lower computation and communication costs as well as lower prediction errors. Extensive experimental evaluations show that FedAAS improves test accuracy, while saving significant communication and computation costs.

ACKNOWLEDGMENTS

The work is supported, in part, by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 (RS21/20), the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-019); and the RIE 2020 Advanced Manufacturing and Engineering (AME) Programmatic Fund (No. A20G8b0102), Singapore.

REFERENCES

[1] Fahao Chen, Peng Li, Toshiaki Miyazaki, and Celimuge Wu. 2021. Fedgraph: Federated graph learning with intelligent sampling. *IEEE Transactions on Parallel and Distributed Systems* 33, 8 (2021), 1775–1786.

[2] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247* (2018).

[3] Jianfei Chen, Jun Zhu, and Le Song. 2017. Stochastic training of graph convolutional networks with variance reduction. *arXiv preprint arXiv:1710.10568* (2017).

[4] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 257–266.

[5] Hanjun Dai, Zornitsa Kozareva, Bo Dai, Alex Smola, and Le Song. 2018. Learning steady-states of iterative algorithms over graphs. In *International conference on machine learning*. PMLR, 1106–1114.

[6] Pan Deng, Xuefeng Liu, Jianwei Niu, and Chunming Hu. 2023. GraphFed: A Personalized Subgraph Federated Learning Framework for Non-IID Graphs. In *2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*. IEEE, 227–233.

[7] Bingqian Du and Chuan Wu. 2022. Federated Graph Learning with Periodic Neighbour Sampling. In *2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*. IEEE, 1–10.

[8] Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2023. Benchmarking graph neural networks. *Journal of Machine Learning Research* 24, 43 (2023), 1–48.

[9] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2021. Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875* (2021).

[10] Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. 2022. Long range graph benchmark. *Advances in Neural Information Processing Systems* 35 (2022), 22326–22340.

[11] Davide Falessi, Aalok Ahluwalia, and Massimiliano DI Penta. 2021. The impact of dormant defects on defect prediction: A study of 19 apache projects. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 1 (2021), 1–26.

[12] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[13] Matthias Fey, Jan E Lenssen, Frank Weichert, and Jure Leskovec. 2021. Gnnautoscale: Scalable and expressive graph neural networks via historical embeddings. In *International Conference on Machine Learning*. PMLR, 3294–3304.

[14] Saeed Ghadimi and Guanghui Lan. 2013. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization* 23, 4 (2013), 2341–2368.

[15] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 1263–1272.

[16] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[17] Andrew Hard, Kanishka Rao, Rajiv Mathews, and Ramaswamy. 2018. Federated learning for mobile keyboard prediction. *DeepAI* (2018).

[18] Chaoyang He, Keshav Balasubramanian, Emir Ceyani, Carl Yang, Han Xie, Lichao Sun, Lifang He, Liangwei Yang, Philip S Yu, Yu Rong, et al. 2021. Fedgraphnn: A federated learning system and benchmark for graph neural networks. *arXiv preprint arXiv:2104.07145* (2021).

[19] Steffen Herbold, Alexander Trautsch, and Jens Grabowski. 2018. A comparative study to benchmark cross-project defect prediction approaches. In *Proceedings of the 40th International Conference on Software Engineering*. 1063–1063.

[20] Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. 2018. Adaptive sampling towards fast graph representation learning. *Advances in neural information processing systems* 31 (2018).

[21] Angelos Katharopoulos and François Fleuret. 2018. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*. PMLR, 2525–2534.

[22] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[23] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[24] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a social network or a news media?. In *Proceedings of the 19th international conference on World wide web*. 591–600.
[25] Anran Li, Lan Zhang, Junhao Wang, Juntao Tan, Feng Han, Yaxuan Qin, Nikolaos M Freris, and Xiang-Yang Li. 2021. Efficient Federated-Learning Model Debugging. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 372–383.
[26] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. 2022. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 965–978.
[27] Rui Liu, Pengwei Xing, Zichao Deng, Anran Li, Cuntai Guan, and Han Yu. 2022. Federated graph neural networks: Overview, techniques and challenges. *arXiv preprint arXiv:2202.07256* (2022).
[28] Xin Liu, Mingyu Yan, Lei Deng, Guoqi Li, Xiaochun Ye, and Dongrui Fan. 2021. Sampling methods for efficient training of graph convolutional networks: A survey. *IEEE/CAA Journal of Automatica Sinica* 9, 2 (2021), 205–234.
[29] Brendan McMahan, Eider Moore, and Ramage. 2017. Communication-efficient learning of deep networks from decentralized data. In *ICML*.
[30] Qiying Pan and Yifei Zhu. 2022. FedWalk: Communication Efficient Federated Unsupervised Node Embedding with Differential Privacy. *arXiv preprint arXiv:2205.15896* (2022).
[31] Morteza Ramezani, Weilin Cong, Mahmut T Kandemir, and Anand Sivasubramaniam. 2021. Learn locally, correct globally: A distributed algorithm for training graph neural networks. *arXiv preprint arXiv:2111.08202* (2021).
[32] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. 2022. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems* 35 (2022), 14501–14515.
[33] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. 2020. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems* 33 (2020), 12559–12571.
[34] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.
[35] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).
[36] Mengying Sun, Sendong Zhao, Olivier Elemento, Jiayu Zhou, and Fei Wang. 2020. Graph convolutional networks for computational drug development and discovery. *Briefings in bioinformatics* 21, 3 (2020), 919–935.
[37] Yue Tan, Yixin Liu, Guodong Long, Jing Jiang, Qinghua Lu, and Chengqi Zhang. 2022. Federated Learning on Non-IID Graphs via Structural Knowledge Sharing. *arXiv preprint arXiv:2211.13009* (2022).
[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
[39] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. 2020. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of The Web Conference 2020*. 1082–1092.
[40] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 28.
[41] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 974–983.
[42] Mikhail Yurochkin, Mayank Agarwal, Nghia Hoang, and Yasaman Khazaeni. 2019. Bayesian nonparametric federated learning of neural networks. In *International conference on machine learning*. PMLR, 7252–7261.
[43] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2019. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931* (2019).
[44] Huanding Zhang, Tao Shen, Fei Wu, Mingyang Yin, Hongxia Yang, and Chao Wu. 2021. Federated Graph Learning–A Position Paper. *arXiv preprint arXiv:2105.11099* (2021).
[45] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. 2021. Subgraph federated learning with missing neighbor generation. *Advances in Neural Information Processing Systems* 34 (2021), 6671–6682.
[46] Taolin Zhang, Chuan Chen, Yaomin Chang, Lin Shu, and Zibin Zheng. 2022. FedEgo: Privacy-preserving Personalized Federated Graph Learning with Ego-graphs. *arXiv preprint arXiv:2208.13685* (2022).
[47] Difan Zou, Ziniu Hu, Song Jiang, Yizhou Sun, and Quanquan Gu. 2019. Layer-dependent importance sampling for training deep and large graph convolutional networks. *Advances in neural information processing systems* 32 (2019).

Received October 2023; revised January 2024; accepted February 2024