A OUTLINE

The appendix is organized as follows:

A.1 Dataset Description

In Graph-FL, we conduct experiments on the compounds networks (MUTAG, BZR, COX2, DHFR, PTC-MR, AIDS, NCI1, hERG) [13, 22, 31, 73, 78, 82], protein networks (ENZYMES, DD, PROTEINS) [7, 15, 31], collaboration network (COLLAB) [42], movie networks (IMDB-BINARY, IMDB-MULTI) [97], super-pixel networks (MNISTSuper-Pixels) [65], and point cloud networks (ShapeNet) [103]. Notably, the node features in Graph-FL often consist of node attributes and labels. In most graph-level GNNs, these components are typically concatenated and fed into the model. Therefore, we report the feature dimension by concatenating node attributes and labels. Notably, super-pixel networks represent images as graphs, with nodes as super-pixels and edges capturing spatial or semantic relationships, enabling effective graph-based classification by leveraging image structure. Detailed dataset description is as follows:

MUTAG [13] is a widely used bioinformatics dataset consisting of 188 graphs, each representing a nitro compound. The nodes are labeled with one of 7 distinct node labels. The primary objective of this dataset is to classify each graph to determine whether the corresponding compound is mutagenic, specifically distinguishing between aromatic and heteroaromatic compounds.

BZR [78] is a bioinformatics dataset used for compound activity prediction, with a primary focus on a collection of benzimidazole compounds. The dataset is designed to indicate the concentration of each compound necessary to inhibit the activity of specific biomolecules, providing valuable insights into the effectiveness of these compounds in biological processes.

COX2 [78] is a dataset centered on Cyclooxygenase-2, an enzyme that plays a critical role in inflammation and pain mechanisms. The dataset is utilized to classify various compounds and predict their potential inhibition potency against the COX-2 enzyme, which is a key target in drug development for anti-inflammatory therapies.

DHFR [78] describes the inhibitory activity of compounds against dihydrofolate reductase, an enzyme critical in cellular folate metabolism and a target for many anticancers. By representing compound molecules in the DHFR dataset as graphs, researchers can employ machine learning models to predict the potential inhibitory effects of unknown compounds against dihydrofolate reductase.

PTC-MR [31] is a bioinformatics dataset consisting of 344 graphs, each representing a chemical compound. Within these graphs, nodes are labeled with one of 19 distinct node labels. The primary objective of this dataset is to predict the carcinogenicity of each compound in rodents, making it valuable for studies related to chemical toxicity and safety assessments.

AIDS [73] is a graph dataset comprising 2000 graphs, each representing molecular compounds derived from the AIDS Antiviral Screen Database of Active Compounds. The dataset includes a total of 4395 chemical compounds, categorized into three classes: 423 compounds belonging to class CA, 1081 to class CM, and the remaining compounds to class CI. This dataset is widely used for tasks involving molecular classification and drug discovery research.

NCI1 [82] is a bioinformatics dataset comprising 4,110 graphs representing chemical compounds. It contains data published by the National Cancer Institute (NCI). Each node is assigned with one of 37 discrete node labels. The graph classification label is determined by NCI anti-cancer screens assessing the ability to suppress or inhibit the growth of a panel of human tumor cell lines.

hERG [22] consists of molecular graphs that represent atoms and chemical bonds within various compounds. This dataset is crucial for predicting the inhibitory effects of these compounds on the human Ether-à-go-go-Related Gene (hERG) potassium channel, which is important in drug safety assessment. The prediction task associated with hERG is essentially a graph regression, making it a valuable resource for studies in drug discovery and toxicology.

ENZYMES [7] is a comprehensive dataset containing 600 protein tertiary structures, meticulously curated from the BRENDA enzyme database. Within the ENZYMES dataset, researchers can explore the intricate structures of six unique enzymes, providing a rich resource for computational analysis and machine learning applications.

DD [15] is a bioinformatics dataset composed of 1,178 graph structures representing proteins. In these graphs, nodes correspond to amino acids, and edges connect nodes that are within 6 Angstroms of each other, reflecting the spatial proximity of amino acids within the protein structure. The primary task associated with this dataset is a binary classification to differentiate between enzymes and non-enzymes, making it a valuable resource for studies in protein function prediction and structural bioinformatics.

PROTEINS [31] is a bioinformatics dataset comprising 1,113 structured proteins. Nodes in these graph-based proteins denote secondary structure elements and are assigned discrete node labels indicating whether they represent a helix, sheet, or turn. Edges indicate adjacency along the amino-acid sequence or in space between two nodes. The objective is to predict the protein function.

COLLAB [42] is a scientific collaboration dataset consisting of 5,000 ego networks represented as graphs. This dataset is compiled from three public collaboration datasets. Each ego network comprises researchers from various fields and is labeled according to the corresponding field, namely High Energy Physics, Condensed Matter Physics, and Astrophysics.

IMDB-BINARY [97] is a movie collaboration dataset comprising 1,000 graphs representing ego networks for actors and actresses. Derived from collaboration graphs within the Action and Romance genres, each graph features nodes representing actors/actresses and edges denoting their collaboration in the same movie. Graphs are labeled according to the corresponding genre, and the objective is to classify the genre for each graph.

IMDB-MULTI [97] is the multi-class extension of the IMDB-BINARY dataset, comprising 1,500 ego-networks. It includes three additional movie genres: Comedy, Romance, and Sci-Fi, making it suitable for multi-class classification tasks. This dataset is commonly used to evaluate the performance of graph-level algorithms.

MNISTSuperPixels [65] is a super-pixel network dataset derived from the original MNIST dataset, where the standard 28x28 pixel images are converted into graphs with 75 nodes each. In this transformation, each node represents a super-pixel, which is a cluster of nearby pixels grouped based on their intensity and spatial proximity. The edges between nodes capture the spatial relationships between these super-pixels.

ShapeNet [103] is a comprehensive point cloud dataset consisting of approximately 17,000 3D shape point clouds spanning 16 different shape categories. Each category is further annotated with 2 to 6 parts, providing detailed segmentation labels. The primary objectives for utilizing this dataset are point cloud classification and segmentation, making it an essential resource for benchmarking algorithms in 3D shape analysis, recognition, and understanding.

As for the Subgraph-FL, OpenFGL integrates citation networks (Cora, Citeseer, PubMed, FedDBLP, ogbn-arxiv) [32, 88, 101], co-purchase networks (Amazon-Computers, Amazon-Photo, ogbn-products) [32, 75], co-author networks (Co-author CS, Co-author Physics) [75], wiki-page networks (Chameleon, ChameleonFilter, Squirrel, SquirrelFilter) [68, 70], actor network (Actor) [68], game synthetic network (Minesweeper) [70], crowd-sourcing network (Tolokers) [70], article syntax network (Roman-empire) [70], rating network (Amazon-rating) [70], social network (Questions) [70], and point cloud networks (PCPNet, S3DIS) [4, 27].

Remarkably, while recent research has highlighted potential data leakage issues due to duplicates in the original Chameleon and Squirrel datasets, considering that previous studies commonly utilized the original versions for validation, we integrated both versions in OpenFGL to offer comprehensive evaluation. Furthermore, point cloud datasets are utilized for downstream tasks where each point is associated with geometric attributes like surface normals and curvature, serving as node features in the graph representation. For surface normal estimation, GNNs are employed to treat each point as a node in the k-nearest neighbors (KNN) graph, learning to predict normal vectors by aggregating information from neighboring points. Similarly, in curvature estimation, GNNs capture local geometric features to predict curvature values. For point cloud classification, the entire point cloud is represented as a graph, with GNNs aggregating point-level information to classify the cloud. In point cloud segmentation, GNNs assign labels to each node, segmenting the cloud based on local and global context. These methods effectively utilize GNNs to leverage the geometric relationships within point clouds for tasks like regression, classification, and segmentation. Developers can flexibly use the above point cloud data. The detailed description of Subgraph-FL datasets is listed below:

Cora, **CiteSeer**, and **PubMed** [101] are widely used citation network datasets, where nodes represent papers and edges denote citation relationships. Node features are word vectors, indicating the presence or absence of specific words in each paper. These datasets are frequently used for node classification.

FedDBLP [88] is the first collected dataset in a distributed manner, where each node represents a published paper and each edge signifies a citation. The bag of words from each paper's abstract is used as node attributes, and the paper's theme is designated as its label. To simulate scenarios where a venue or organizer restricts citations of its papers, users can split the dataset based on each node's venue or the organizer of that venue.

ogbn-arxiv [32] is a widely used citation graph indexed by Microsoft Academic Graph (MAG) [85], especially for the large-scale graph learning. Each paper in the dataset is represented by the average of the word embeddings derived from its title and abstract. These word embeddings are generated using the skip-gram model, which captures semantic relationships between words based on their context within the text. This dataset is widely used for graph-based learning tasks, such as node classification.

Amazon Photo and **Amazon Computers** [75] are subsets of the Amazon co-purchase graph, where nodes represent individual products, and edges signify that two products are frequently bought together. The node features for these datasets are derived from product reviews, represented as bag-of-words vectors, capturing the textual information associated with each item. These datasets are commonly used for graph-based downstream tasks such as node classification in graph-based recommendation systems.

ogbn-products [32] is a co-purchasing network where nodes represent products and edges indicate frequent co-purchases. The node features are derived from bag-of-words representations of product descriptions. Due to its extensive size and complex structure, this dataset is particularly well-suited for large-scale graph learning applications, making it an ideal benchmark for evaluating the scalability and performance of graph-based algorithms.

Coauthor CS and **Coauthor Physics** [75] are co-authorship graphs derived from the MAG [85]. In these graphs, nodes represent individual authors, edges denote co-authorship relationships between them, and node features are constructed from the keywords of the authors' publications. The labels assigned to the nodes indicate the specific research fields in which the authors are active. These datasets are commonly used for evaluating graph-based methods, particularly in the context of node classification.

Chameleon and **Squirrel** [68] are two page-page networks extracted from specific topics within Wikipedia. In these datasets, nodes represent web pages, while edges signify mutual links between pages. Node features are derived from several informative nouns found on Wikipedia. They categorize the nodes into five groups based on the average monthly web page traffic.

Chameleon Filter and **Squirrel Filter** [70] emphasis nodes in original datasets share the same regression target and neighborhood simultaneously, resulting in duplicates. These duplicates are present across the training, validation, and test sets, causing data leakage. Therefore, these filtered versions enable a fairer comparison.

Actor [68] is an actor co-occurrence network where nodes represent actors, and edges indicate their co-appearance on Wikipedia pages. Node features are bag-of-words vectors derived from these pages, and actors are categorized into five groups based on the terms found in their respective Wikipedia entries. This dataset is commonly used for graph-based tasks like node classification.

Minesweeper [70] draws inspiration from the Minesweeper game and stands as the synthetic dataset. The graph is a regular 100x100 grid, where each node (cell) is linked to its eight neighboring nodes (excluding nodes at the grid's edge, which have fewer neighbors). Twenty percent of the nodes are randomly designated as mines. The objective is to predict which nodes conceal mines. Node features consist of one-hot-encoded counts of neighboring mines. However, for a randomly chosen 50% of the nodes, the features are undisclosed, indicated by a distinct binary feature.

Tolokers [70] is derived from the crowdsourcing platform [56]. Nodes correspond to workers who have engaged in at least one of the 13 selected projects. An edge connects two workers if they have collaborated on the same task. The objective is to predict which workers have been banned in one of the projects.

Roman-empire [70] is based on the Roman Empire article from the English Wikipedia [44], each node corresponds to a non-unique word in the text, mirroring the article's length. Nodes are connected by an edge if the words either follow each other in the text or are linked in the sentence's dependency tree. Thus, the graph represents a chain graph with additional connections.

Amazon-ratings [70] is derived from the co-purchasing network and its metadata available in the SNAP [43]. Nodes are items and edges connect items frequently bought together. The task is predicting the average rating given by reviewers, categorized into five classes. Node features are based on the FastText embeddings [24] of words in the product description. To manage graph size, only the largest connected component of the 5-core is considered.

Questions [70] is derived from data collected from the question-answering platform Yandex Q. In this dataset, nodes represent users, and an edge exists between two nodes if one user answers another user's question within a one-year timeframe (from September 2021 to August 2022). The objective is to predict which users remained active on the website (i.e., were not deleted or blocked) by the end of the specified period. For node features, it utilizes the average FastText embeddings for words found in the user descriptions.

PCPNet [27] is a point cloud dataset consisting of 30 distinct shapes, each represented as a densely sampled point cloud with 100,000 points. For each shape, surface normals and local curvatures are provided as node features, capturing essential geometric properties. This dataset is intended for tasks such as point cloud classification and segmentation, making it a valuable resource for evaluating algorithms in graph learning and 3D shape analysis.

S3DIS [4] is a point cloud dataset comprising six large-scale indoor areas from three different buildings. It includes 12 semantic elements, such as walls, floors, and furniture, as well as one clutter class, making it a diverse dataset for indoor scene understanding. The primary objectives for using this dataset are point cloud classification and segmentation, offering a challenging benchmark for evaluating the performance of algorithms in recognizing and segmenting complex indoor environments.

A.2 Baseline Description

Given the uniqueness of FGL, the baseline of OpenFGL consists of three components: (1) Backbone graph learning models in multi-clients; (2) Prevalent FL algorithm in graph-independent scenarios (i.e., computer vision); (3) Recently proposed FGL algorithms.

For (1), considering that most FGL approaches entail additional design for the local backbone model, we have implemented only the most popular baseline models (GCN [40], GAT [81], GraphSAGE [28], SGC [90], GCNII [10], GIN [95], TopKPooling [19], SAGPooling [41], EdgePooling [14], and PANPooling [62]) in centralized graph learning, which is generally applicable to both graph-FL and subgraph-FL scenarios, providing flexibility to the future FGL developers. The backbone GNN details implemented in our proposed OpenFGL are listed below:

GCN [40] introduces a novel approach to graphs that is based on a first-order approximation of spectral convolutions on graphs. This approach learns hidden layer representations that encode both local graph structure and features of nodes.

GAT [81] utilizes attention mechanisms to quantify the importance of neighbors for message aggregation. This strategy enables implicitly specifying different weights to different nodes in a neighborhood, without depending on the graph structure upfront.

GraphSAGE [28] leverages neighbor node attribute information to efficiently generate representations. This method introduces a general inductive framework that leverages node feature information to generate node embeddings for previously unseen data.

SGC [90] simplifies GCN by removing non-linearities and collapsing weight matrices between consecutive layers. Theoretical analysis show that the simplified model corresponds to a fixed low-pass filter followed by a linear classifier.

GCNII [10] incorporates initial residual and identity mapping. Theoretical and empirical evidence is presented to demonstrate how these techniques alleviate the over-smoothing problem.

GIN [95] construct a straightforward architecture that is demonstrably the most expressive within the GNN class and matches the power of the Weisfeiler-Lehman graph isomorphism test.

MeanPooling [95] is a parameter-free pooling operation in graph neural networks. It generates a graph embedding by averaging the all node embeddings, encoding the complex graph into a unified vector representation.

TopKPooling [19] introduces novel graph pooling and unpooling operations. The former adaptively selects nodes to form a smaller graph based on their scalar projection values on a trainable projection vector. The latter, as the inverse operation, restores the graph to its original structure using the positional information of nodes selected in the corresponding pooling layer.

SAGPooling [41] proposes a graph pooling approach based on self-attention. By utilizing self-attention with graph convolution, this method considers both node features and graph topology.

EdgePooling [14] proposes a graph pooling layer based on edge contraction. This strategy learns a localized and sparse pooling transform to improve predictive performance. It can be integrated into existing GNN architectures without adding any additional losses or regularization.

PANPooling [62] utilizes a convolution operation that considers every path linking the message sender and receiver, with learnable weights dependent on the path length, corresponding to the maximal entropy random walk. This strategy offers a versatile framework adaptable to different graph data sizes and structures.

Regarding (2), while some prevalent FL approaches (FedAvg [63], FedProx [48], Scaffold [37], MOON [47], and FedDC [20]) have demonstrated effectiveness in computer vision-based (CV-based) federated scenarios, recent FGL studies contend that they only achieve sub-optimal performance. This weakness is attributed to their failure to incorporate topological information during the collaborative optimization process of FGL. Therefore, we only implement prevalent FL as baselines to assist future FGL developers in evaluating the effectiveness of their proposed methods.

Additionally, heterogeneity has been a persistent challenge in FL, encompassing multi-level heterogeneity arising from different local systems. Specifically, this multi-level heterogeneity in terms of: (1)

Data heterogeneity arises from variations in local data collection methods and data quality, leading to optimization challenges due to diverse multi-source data characterized by Non-iid data and domain shift; (2) Model heterogeneity stems from varying computational resource requirements, scalability needs, and predictive performance criteria across local systems, prompting the adoption of distinct local backbone models; (3) Communication heterogeneity is driven by variations in communication bandwidth among local devices, demanding the minimization of communication overhead. Nonetheless, to enhance predictive performance in collaborative training, FL algorithms frequently necessitate increased information sharing, either at the client level or between clients and servers.

Building upon this foundation, to provide effective baselines, we have particularly implemented prototype-based methods (FedProto [80], FedNH [11], FedTGP [107]). This has sparked a research trend in recent CV-based FL, as this technology can comprehensively address the aforementioned multi-level heterogeneity challenges. The prevalent graph-independent FL baseline details implemented in our proposed OpenFGL are listed below:

FedAvg [63] serves as a foundational method in FL, enabling decentralized model training across diverse devices while preserving data privacy. Initiated by a central server that distributes a global model, clients independently execute local updates through stochastic gradient descent. Subsequently, these updates are aggregated by the server via averaging to refine the global model, with the cycle repeating until convergence.

FedProx [48] allows for variable amounts of work to be performed locally across devices, and relies on a proximal term in model align loss to help stabilize the method. Theoretically, it offers convergence guarantees under conditions of non-identical data distributions and variable device workloads.

Scaffold [37] employs control variates to mitigate client-drift in FL. Demonstrating significant reductions in communication rounds, Scaffold is resilient to data heterogeneity and client sampling.

MOON [47] is a model-contrastive FL framework that enhances local training by leveraging model representation similarities through contrastive learning at the model level.

FedDC [20] is a novel FL algorithm that corrects local drift through lightweight modifications. Each client tracks the deviation between local and global model parameters using an auxiliary variable, enhancing parameter-level consistency.

FedProto [80] is the first federated prototype learning framework for FL heterogeneity. Instead of exchanging gradients, clients and the server share abstract class prototypes. FedProto aggregates local prototypes and distributes global ones back to clients to regularize local model training, aiming to align local prototypes with global standards while minimizing local classification errors.

FedNH [11] addresses class imbalance by enhancing both personalization and generalization of local models. FedNH distributes class prototypes uniformly in the latent space, infusing class semantics to prevent prototype collapse and enhance model performance. This dual approach improves local models, boosting the generalization of the global model and thus refining personalized models.

FedTGP [107] unlikes conventional methods that aggregate prototypes via weighted averaging, FedTGP uses adaptive contrastive Learning to train global prototypes on the server, enhancing prototype separability and preserving semantic integrity.

As for (3), it is the core of OpenFGL as a comprehensive benchmark. To provide future FGL researchers with a comprehensive testing library and a user-friendly development framework, we conducted a thorough review of recent FGL studies, encompassing both the Graph-FL (GCFL+ [93], FedStar [79]) and Subgraph-FL (FedSage+ [110], Fed-PUB [5], FedGTA [52], FGSSL [34], FedGL [8], AdaFGL [51], FGGP [83], FedTAD [115], FedDEP [108]) scenarios, and comprehensively integrated them. The FGL baseline details implemented in our proposed OpenFGL are listed below:

GCFL+ [93] dynamically clusters local systems using GNN gradient patterns to reduce structural and feature heterogeneity, particularly in the Graph-FL scenarios. Addressing the issue of fluctuating gradients, they enhance GCFL with a gradient sequence-based clustering mechanism using dynamic time warping, thereby improving clustering quality and theoretical robustness.

FedStar [79] shares structural embeddings across clients using an independent structure encoder. This design allows FedStar to capture domain-invariant structural information while enabling personalized feature learning, thereby avoiding feature misalignment and enhancing inter-graph learning efficacy.

FedSage+ [110] integrates node features, link structures, and labels using a GraphSage model and FedAvg across local subgraphs. FedSage+ extends this by training a generator to address missing links, enhancing model robustness and completeness.

Fed-PUB [5] is a novel framework for personalized subgraph FL that enhances local GNNs interdependently rather than forming a single global model. Fed-PUB computes similarities between local GNNs using functional embeddings derived from random graph inputs, facilitating weighted averaging for server-side aggregation. Additionally, it employs a personalized sparse mask at each client to selectively update subgraph-relevant parameters.

FedGTA [52] innovatively merges large-scale graph learning with FGL. Clients encode topology and node attributes, compute local smoothing confidence and mixed moments of neighbor features, and then upload these to the server. The server uses this data to perform personalized model aggregation, utilizing local smoothing confidence as weights for effective integration.

FGSSL [34] handles local client distortion in FL by focusing on node-level semantics and graph-level structures via the well-designed contrastive loss functions. They enhance node discrimination by aligning local nodes with their global counterparts of the same class and distancing them from different classes. Additionally, FGSSL transforms adjacency relationships into similarity distributions, using the global model to distill relational knowledge into local models, preserving both structure and discriminability.

FedGL [8] identifies global self-supervision information, which is then utilized to enhance prediction accuracy. Specifically, FedGL involves uploading prediction outcomes and node embeddings to the server to derive global pseudo labels and a global pseudo graph. These global insights are distributed to each client, augmenting training labels and refining graph structures, consequently enhancing the performance of local models.

AdaFGL [51] introduces a two-step personalized approach that first aggregates multi-client models into a federated knowledge extractor during the final round at the server. Subsequently, each client undertakes personalized training utilizing the local subgraph and this federated extractor.

FGGP [83] divides the global model into two tiers linked by prototypes. At the classifier level, FGGP replaces traditional classifiers with clustered prototypes to enhance class discrimination and multi-domain prediction accuracy. Meanwhile, at the feature extractor level, FGGP leverages contrastive learning to imbue prototypes with global knowledge, thereby improving model generalization.

FedTAD [115] initially computes topology-aware node embeddings to evaluate the reliability of class-wise knowledge, transmitting this information to the server. Guided by the class-wise knowledge reliability, FedTAD on the server side conducts data-free knowledge distillation to transfer reliable knowledge from local models across multiple clients to the global model.

FedDEP [108] leverages GNN embeddings for deep neighbor generation based on the FedSage+, employing efficient pseudo-FL for neighbor generation through embedding prototyping, and ensuring privacy protection via noiseless edge local DP. Meanwhile, it utilizes prototype representation technologies to further reduce communication costs.

A.3 Metric Description

Given the diverse downstream tasks in the FGL scenarios, we implement the following evaluation metrics tailored for regression (MSE, RMSE), classification (Accuracy, Precision, Recall, F1), prediction (AUC-ROC, AP), and clustering (Clustering-accuracy, NMI, ARI) tasks. Notably, in link prediction tasks within graph machine learning, AUC-ROC and AP are preferred over accuracy because they better handle the typical class imbalance, where non-existent links far outnumber actual links. Accuracy can be misleading, as a model might achieve high accuracy by simply predicting the majority class. In contrast, AUC-ROC and AP focus on the model's ability to correctly rank positive links higher than negative ones, providing a more reliable evaluation of performance in scenarios where correct identification of the minority class (actual links) is crucial.

Mean Squared Error is a prevalent metric in graph regression tasks. It measures the average squared deviation between predicted values and actual ground truth across the entire dataset. Lower MSE values signify superior model performance, indicating a tighter alignment of predicted outcomes with observed results.

Root Mean Squared Error is an extension of MSE and offers a more interpretable measure by taking the square root of the average squared differences between predicted and actual values. Like MSE, lower RMSE denotes better alignment.

Accuracy stands as a foundational metric in classification tasks, quantifying the ratio of correctly classified instances to the total instances in a dataset. It offers a clear indication of a graph learning model's overall predictive capability. Higher accuracy values reflect a better alignment between predicted and actual class labels, demonstrating the model's effectiveness.

Precision focuses on positive predictions. Unlike Accuracy, it emphasizes the correctness of positive predictions by measuring the ratio of correctly predicted positive samples to all predicted positive samples. This aspect becomes particularly critical when dealing with imbalanced datasets.

Recall measures a model's ability to capture all positive instances. Unlike Precision, it emphasizes correctly identified positive samples and overall actual positives. This metric is vital in scenarios where missing positives have critical implications.

F1 Score represents the harmonic mean of Precision and Recall. This metric provides a balanced assessment of a model's performance by considering both the precision of positive predictions and the model's ability to capture all positive instances. F1 Score is particularly valuable in scenarios where achieving high precision and recall are equally important.

Area Under the Receiver Operating Characteristic curve quantifies the performance of a model in distinguishing between positive and negative links. It provides a comprehensive measure of a model's ability to rank positive links higher than negative ones. The high AUC-ROC indicates that the model effectively discriminates between positive and negative links.

Average Precision measures the quality of a model's ranked list of positive links by calculating the average precision at each relevant position. Unlike AUC-ROC, AP focuses solely on the precision-recall curve, providing a more detailed assessment of a model's performance, especially in imbalanced datasets where positive links are rare. The high AP indicates that the model effectively ranks positive links higher than negative ones.

Clustering-accuracy measures the agreement between the cluster assignments produced by a clustering algorithm and a ground truth clustering. This metric differs from traditional accuracy metrics used in node classification, as it evaluates the overall coherence of cluster assignments rather than individual node labels. A higher clustering accuracy indicates a better alignment between the identified clusters and the true underlying structure of the graph, thus reflecting the effectiveness of the clustering algorithm in uncovering meaningful communities or groups of nodes.

Normalized Mutual Information quantifies the similarity between predicted clusters and ground truth by measuring the mutual information while normalizing for cluster size imbalances. NMI ranges from 0 to 1, where higher values indicate better agreement between the predicted clusters and ground truth. This metric is particularly valuable in scenarios where accurately identifying community structures or functional groups within a graph is critical.

Adjusted Rand Index quantifies the similarity between predicted clusters and ground truth while considering the chance-corrected agreement. ARI ranges from -1 to 1, where values closer to 1 indicate better agreement between the predicted clusters and ground truth than random clustering.

A.4 Robustness Simulation Description

Given the practical applications driving FGL studies, the pivotal goal of various FGL approaches should be their effective deployment in real-world industrial scenarios. Consequently, conducting a thorough evaluation of the robustness of existing methods becomes essential. In our proposal, besides exploring the generalization of current methods across various federated data simulation settings as discussed in Sec. 3.1, we draw insights from common business challenges encountered in industrial scenarios. Specifically, we additionally integrate the following experimental setups to provide a comprehensive evaluation for industrial research projects from a robustness perspective.

Feature Noise. In real-world FGL applications, such as healthcare and finance, distributed privacy data is independently collected by local agents, leading to variations in data collection methods, processing techniques, and data sources. These variations contribute to node feature noise, a common issue in practical settings. For example, in healthcare, differences in medical equipment, patient demographics, and data entry practices across hospitals can result in inconsistent patient data, introducing noise into the node features representing these data points. In finance, variations in transactional systems, data aggregation methods, and processing protocols across institutions can lead to discrepancies in financial data, further contributing to node feature noise. To accurately evaluate the robustness of existing FGL methods under such conditions, we simulate this scenario by introducing Gaussian or Laplacian noise into the node features of data samples within each client. Notably, Gaussian noise reflects natural data collection fluctuations, while Laplacian noise captures more pronounced deviations.

Edge Noise. In our implementation, we introduce two edge perturbation approaches: heterophilous noise (Subgraph-FL) and meta noise (Subgraph-FL and Graph-FL). For heterophilous noise, we randomly select non-connected node pairs for heterophilous perturbations based on their labels. This approach is motivated by recent studies [58, 61, 113], which indicate that most GNNs struggle with heterophilous topology. Despite some GNN designs mitigating this issue, it has been rarely addressed in most FGL studies. Moreover, heterophily is prevalent in the real world despite homophilous topology presumably dominating in default. As for meta noise, generated by Metattack [116], we budget the attack as 0.2 for each local dataset. This approach shares the motivation of heterophilous noise but represents a more generalized and sophisticated perturbation method for both two FGL scenarios, which achieves optimal noise injection mechanisms through learnable means. In real-world applications, such as social networks, varying user interaction patterns across platforms can lead to inaccurate connections between nodes. These discrepancies create noise edges, reflecting the inherent challenges of decentralized data environments.

Label Noise. Due to the diversity in data collection methods and the quality of local data sources, label noise is prevalent in crowd-sourcing scenarios. In this scenario, the update process of the local model is inevitably affected, resulting in perturbed models. When these model weights are uploaded to the server for multi-client collaboration, the naive federated paradigm suffers from global knowledge confusion, thereby significantly impacting the initialization of local models for the next round. To simulate this setting, we introduce a novel perspective for evaluating algorithm robustness by randomly perturbing the true labels of training set samples according to a certain proportion. For instance, in crowd-sourcing, workers from diverse backgrounds may label the same data differently, leading to inconsistencies. These factors contribute to prevalent label noise, which disrupts the local model training process, resulting in perturbed models. When these models are aggregated in the federated collaboration, the naive model aggregation mechanism struggles with global knowledge confusion, ultimately affecting the initialization of local models in the next training round. To simulate this real-world challenge, we introduce a novel approach to evaluate algorithm robustness by randomly perturbing the true labels of training set samples according to a specified proportion.

Feature/Edge/Label Sparsity. In the current data explosion era, gathering substantial volumes of high-quality data can incur significant economic costs. Additionally, the laborious annotation requests both substantial manual labor and computational resources and leads to the prevalence of sparse data. In graph-structured data, this sparsity challenge often manifests in missing attributes in feature dimensions, sparse graphs, and the well-known label sparsity issue. To integrate the aforementioned scenarios into our proposed OpenFGL framework and evaluate the robustness of existing FGL studies from an industrial application perspective, we provide the following implementation details: In the feature sparsity setting, we assume that the feature representation of unlabeled nodes is partially missing. To simulate edge sparsity, we randomly remove edges from subgraphs, providing a more challenging but realistic scenario. For label sparsity, we change the ratio of labeled nodes.

Client Active Fraction/Client Sparsity. In practical FGL scenarios, it is necessary to select a subset of clients to participate in each round to reduce communication costs or unavoidable device dropouts. However, the reduction in the number of clients participating in collaborative training during each communication round may lead to the global model deviating from the global optima. In such a setting of *client sparsity*, it is crucial to test whether FGL algorithms have the capability to accurately locate global optima through sparse data distribution. Notably, client sparsity is a unique yet highly significant perspective for evaluating algorithm robustness in federated distributed scenarios.

Based on the above noise and sparsity setting, we can evaluate the resilience and performance of FGL studies under conditions of data perturbation and practical scenarios, offering insights into their robustness in real-world applications where data and deployment environment may be imperfect.

A.5 Effectiveness Evaluation Strategies

During our investigation, we observer a lack of descriptions for evaluating the effectiveness of existing FGL studies. Therefore, in this section, we aim to present structured criteria from both data and model perspectives. These criteria aim to standardize the effectiveness evaluation for future FGL studies and support the experimental settings of this paper.

To begin with, due to privacy regulations and prohibitively high manual costs, similar to FL in the computer vision domain, most existing FGL studies adopt a data partition strategy based on global data (i.e., benchmark datasets under centralized evaluation) to simulate distributed scenarios. Based on this, we allocate the partitioned global data as multiple sets of local private data to different clients. Therefore, with the aforementioned data-driven experimental settings, we gain access to both implicit global data and local data from each client, providing an aspect for algorithm evaluation. Subsequently, in federated collaborative training, most FGL algorithms entail local training on private data at each client and model aggregation at the server side. This procedure gives rise to two perspectives for evaluating the models produced by the algorithms: (1) server-side global model, typically transmitted from local models to the server and refined through well-designed server-side model aggregation or update mechanisms. Since it integrates most local models from the current communication round, we refer to it as the global model. (2) client-side local model, mainly updated by local private data, which, in comparison to the global model, emphasizes fitting local data more closely. It is often emphasized by personalized algorithms, as they focus on the local training frameworks. To this end, considering both data and model perspectives, we acquire global data and local data, along with the global model and local model, respectively. Combining these aspects results in four effectiveness evaluation strategies in FGL, which reflect the diverse business requirements of practical industrial scenarios. In a nutshell, we derive the following four evaluation criteria:

(1) Global Model on Global Data: This primarily aims to verify the generalization of current FGL algorithms by evaluating the performance of the server-side collaborative model on a broader empirical data domain (i.e., global data). This evaluation is conducted more for experimental analysis from a generalization perspective during the research process.

(2) Global Model on Local Data: This evaluation is designed to facilitate multi-client collaborative training facilitated by a trusted server, leveraging diverse knowledge to enhance the robustness of the server-side collaborative model without direct data sharing. In practical applications, multiple clients undergo standardized training and evaluation under the supervision of a trusted server.

(3) Local Model on Global Data: Similar to (1), this evaluation is aimed at empirically analyzing personalized FGL algorithms from a generalization perspective. Considering that personalized algorithms highlight the distinct neural architectures or learning mechanisms of individual local clients, essential generalization analysis is conducted to determine whether the current algorithm can produce unbiased predictions and mitigate over-fitting issues. This analysis involves evaluating the performance of client-side personalized models on global data.

(4) Local Model on Local Data: This evaluation represents the most application-driven evaluation strategy among the above strategies, as it aligns with the practical requirements of FL. In FL, multiple clients collaborate to enhance their local scenarios with more robust personalized models while addressing privacy concerns. Consequently, in our comprehensive benchmark incorporating multiple algorithms, we default to this effective evaluation strategy. This approach, driven by practical applications, serves to verify the real-world deployment feasibility of current algorithms.

A.6 Experiment Environment

The experiments are conducted on the machine with Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz, and NVIDIA A100 80GB PCIe and CUDA 12.2. The operating system is Ubuntu 18.04.6 with 216GB memory. As for software versions in the environment, we use Python 3.9 and Pytorch 1.11.0.

A.7 Hyperparameter Settings

General Experimental Settings. For Graph-FL, the learning rate is typically set to 1×10^{-3}, with each client performing 1 epoch per communication round and a batch size of 128. In Subgraph-FL, the learning rate is raised to 1×10^{-2}, and the local epochs are extended to 3. This adjustment accommodates the typically larger scale of node samples in Subgraph-FL, necessitating a larger learning rate and more local iterations to facilitate model convergence. Additionally, for both scenarios, we standardize certain parameters. The weight decay is set to 5×10^{-4}, the number of communication rounds is set to 100, the dropout rate is set to 0.5, and optimization is conducted using the Adam [39] optimizer. To evaluate the robustness of our results under varying initial conditions, we eliminate the use of fixed random seeds. All experiments are repeated three times to report the mean and variance of the respective metrics for unbiased predictive performance.

Personalized Baseline Settings. We perform extensive hyperparameter tuning to ensure a comprehensive and unbiased evaluation of these FGL methods using the Optuna framework [3]. The hyperparameter search spaces for all baselines are available in our GitHub repository. For detailed explanations of these hyperparameters, please refer to their original papers.

Regarding graph-specific data simulation strategies in Table 5, to enhance readability and avoid complex figures or tables, we default to using 10-client label Dirichlet (i.e., Label Distribution Skew and $\alpha = 1$) and Metis partitioning (i.e., Metis-based Community Split) separately for the Graph-FL and Subgraph-FL scenarios. The former is inspired by data Non-iid simulation in CV [37, 47, 48], while the latter is inspired by prevalent data simulation strategies in current FGL studies [5, 51, 52]. Experimental evaluations of the generalization of existing methods across different data simulation scenarios can be found in Sec. 4.2. Furthermore, in the selection of local backbone models for graph learning, we choose prevalent GIN and GCN models, applied to Graph-FL and Subgraph-FL, respectively. Notably, we experiment with multiple datasets and baselines in separate modules and use graph/node classification to report experimental results to further avoid complex charts, making the results more reader-friendly.

A.8 DP-based Privacy Persevere

A.8.1 Preliminaries on Differential Privacy. DP [16] has become the dominant model for the protection of individual privacy from powerful and realistic adversaries. Informally, it requires that the output of a differentially private query is not dramatically affected by the inclusion or exclusion of any particular individual's data in the input. This means that even if an attacker can access all but one individual's data, they cannot determine whether it was included in the computation. The formal definition of DP is as follows:

Definition A.1 (Differential Privacy [16])**.** Let D and D' be two adjacent datasets that differ in at most one entry. A randomized algorithm $\mathcal{A}$ satisfies (ϵ, δ)-differential privacy if for all $O \subseteq Range(\mathcal{A})$:

$$[\mathcal{A}(D) \in O] \leq e^{\epsilon} \cdot [\mathcal{A}(D') \in O] + \delta. \quad (1)$$

The privacy budget ϵ controls the trade-off between the level of privacy protection and utility: a lower ϵ indicates stricter privacy preservation but leads to lower utility. The parameter δ represents the maximum permissible failure probability and is usually chosen to be much smaller than the inverse of the number of data records.

The formal definition of DP revolves around the concept of adjacency between datasets. When data are represented as a graph, two notions of adjacency are defined: *edge*-level and *node*-level adjacency. Edge-level adjacency occurs when two graphs differ by just one edge, while node-level adjacency involves a difference in an entire node and its associated features, labels, and connections.

Therefore, an algorithm $\mathcal{A}$ provides edge-level (or node-level) (ϵ, δ)-DP if for any two edge-level (or node-level) adjacent graph datasets G and G' and any possible outputs $O \subseteq Range(\mathcal{A})$, the inequality $[\mathcal{A}(G) \in O] \leq e^{\epsilon} \cdot [\mathcal{A}(G') \in O] + \delta$ holds. Edge-level DP focuses on the protection of edge privacy, while node-level DP aims to protect the privacy of nodes and their connections. In consequence, the node-level DP can provide more robust privacy protection.

In this paper, we use an alternative definition of DP, called *Rényi Differential Privacy* (RDP) [64], since it allows for tighter composition of DP across multiple steps.

Definition A.2 (Rényi Differential Privacy [64])**.** An randomized algorithm $\mathcal{A}$ is said to be (α, γ)-RDP, if, for every pair of adjacent datasets G and G', we have

$$D_{\alpha}(\mathcal{A}(G) \| \mathcal{A}(G')) \leq \gamma, \tag{2}$$

where $D_{\alpha}(P\|Q)$ is the Rényi divergence of order α between probability distributions P and Q defined as:

$$D_{\alpha}(P\|Q) = \frac{1}{\alpha - 1} \log \mathbb{E}_{x \sim Q} \left[\frac{P(x)}{Q(x)} \right]^{\alpha}. \tag{3}$$

The concept of RDP is closely related to the standard (ϵ, δ)-DP. According to [64], any mechanism that achieves (α, γ)-RDP also fulfills $(\gamma + \log(1/\delta)/(\alpha - 1), \delta)$-DP for any $\delta \in (0, 1)$. A basic mechanism to achieve RDP is the Gaussian mechanism. Specifically, we inject Gaussian noise into the algorithm's output for privacy protection. And the noise follows the Gaussian distribution $N(0, \alpha\Delta_2^2/2\gamma)$, where Δ_2 represents the ℓ_2-sensitivity.

Definition A.3 (ℓ_2-Sensitivity)**.** Given a function $f\colon \mathcal{G} \to \mathbb{R}^d$, the ℓ_2-sensitivity of f is defined as

$$\Delta_2 = \max_{G,G'} ||f(G) - f(G')||_2, \tag{4}$$

where G and G' are adjacent datasets and $|| \cdot ||_2$ is the ℓ_2 norm.

A.8.2 Privacy-Preserving Techniques in OpenFGL. In federated collaboration, each client uploads model gradients to the server. However, the gradients computed directly from raw data are susceptible to inference attacks [59, 66] and reconstruction attacks [23, 114], which can lead to privacy breaches. OpenFGL implements basic privacy-preserving techniques that satisfy node-level RDP to protect individual privacy when model gradients are uploaded. The core idea is that each client perturbs the gradients via the Gaussian mechanism and then sends a perturbed version to the server.

Specifically, each client receives initial parameters from the server. Then, similar to the standard mini-batch SGD technique, each client samples a subset $\mathcal{S}$ that consists of k samples selected uniformly at random from the training set. Clients can compute the gradients via forward and backward propagation within this mini-batch. Given that there is no a priori constraint on the size of the model gradients, we employ the clipping operator $Clip_C$ to handle the gradient of each sample $\mathbf{w}_i$: $Clip_C(\mathbf{w}_i) = \mathbf{w}_i \cdot \min(1, C/||\mathbf{w}_i||_2)$, where C is the clipping threshold. However, in the context of GNNs, all direct and multi-hop neighbors participate in the calculation of gradients for each node via recursive layer-wise message passing [29]. At each layer, the representation of each node is derived not solely from its features but also from the features of adjacent nodes. Therefore, each per-sample gradient term can be influenced by private data from multiple nodes [12]. This means the sensitivity of the gradient due to the presence or absence of a node can be extremely high due to the node itself and its neighbors, which makes standard DP-SGD-based methods [2] infeasible, resulting in either high privacy cost or poor utility due to the considerable required DP noise. In this paper, we restrict the number of graph convolutional layers and study models with only one GNN layer. Under this limitation, the sensitivity of 1-Layer GNN as follows:

Lemma A.4 (Node-Level Sensitivity of the 1-Layer GNN [12])**.** *For any node v_i, let $\mathbf{y}_i$ represent the ground truth and $\tilde{\mathbf{y}}_i$ the prediction from a 1-layer GNN. Consider the loss function $\mathcal{L}$ of the form: $\mathcal{L}(G, \Theta) = \sum_{v_i \in \mathcal{V}} \ell(\tilde{\mathbf{y}}_i; \mathbf{y}_i)$. The ℓ_2-sensitivity of the aggregated gradient, $\mathbf{w}_G = \sum_{v_i \in \mathcal{S}} Clip_C(\nabla_{\Theta}\ell(\tilde{\mathbf{y}}_i; \mathbf{y}_i))$, is determined by the equation:*

$$\Delta_2(\mathbf{w}_G) = 2(d_{max} + 1)C, \tag{5}$$

where d_{max} denotes the maximum degree of the graph G.

To achieve a better trade-off between privacy and utility, we also utilize the privacy amplification technique [12], which is implemented by sampling. Next, we sample noise from a Gaussian distribution $N(0, \alpha\Delta_2^2(\mathbf{w}_G)/2\gamma)$ and add the noise to the aggregated gradient over the mini-batch. Finally, the perturbed gradients are sent to the server for model aggregation. The server aggregates the perturbed gradients from all clients and updates the global model parameters. The process is repeated until the model converges.

However, when a smaller privacy budget is allocated, there is a significant degradation in performance, as illustrated in Table 9. This decline is primarily attributed to the unique characteristics of the GNN model, as previously mentioned. This uniqueness results in a high sensitivity of the model gradient and affects the privacy amplification techniques. Consequently, substantial noise is introduced into the gradients during training, which markedly disrupts the learning process. This indicates that training graph neural networks in federated scenarios still faces a tough test. Therefore, it is crucial to develop new privacy-preserving techniques that can effectively protect privacy while maintaining performance.

For Graph-FL, where each individual sample represents an entire graph, the application of DP-SGD [2] can be directly extended to these tasks. This approach not only preserves privacy but also optimizes the model effectively, making it particularly suitable for scenarios where maintaining the integrity and confidentiality of the entire graph as a cohesive data unit is crucial. This method is ideal for ensuring that the learning process respects the privacy constraints inherent in sensitive data environments.

A.8.3 Technique Details of FedDEP. In addition, we have implemented the FedDEP [109] algorithm within the OpenFGL framework. FedDEP achieves noise-free edge-level DP by employing random sampling, ensuring robust privacy preservation without compromising data integrity. To tackle the issue of cross-subgraph missing neighbors, FedDEP incorporates an advanced deep neighbor generation module known as DGen, which enhances the model's capability to generate and integrate missing neighbors effectively. To further optimize computational efficiency within each client, local GNN embeddings are clustered to create sets of missing neighbor prototypes. Moreover, to reduce inter-client communication overhead, FedDEP introduces a pseudo-federated learning approach, where these prototype embeddings are shared across the system before the training of DGen, thereby streamlining the collaborative learning process while maintaining model accuracy and privacy.

To protect privacy, FedDEP adapts noise-free differential privacy, originally developed for general domains [77], to edge-level local differential privacy. Specifically, it implements two random sampling strategies during the FedDEP model training: (i) random neighborhood sampling within each graph convolutional layer, and (ii) random sampling of the generated deep neighborhoods by a Bernoulli sampler. These strategies collectively aim to obscure the individual contributions of nodes, which ensures privacy guarantees with the smallest possible impact on the accuracy of the model. However, the FedDEP algorithm is designed for edge-level DP, which only protects the privacy of edge and not node features or node embeddings. This limitation may lead to potential privacy breaches when the node features contain sensitive information.

A.8.4 Future Research Directions. This section introduces several promising research directions in DP-based FGL. Currently, there are few privacy-preserving algorithms specifically tailored for FGL. Given the distinctive properties of graph data, traditional privacy mechanisms commonly employed in FL fail to seamlessly extend to this domain. Therefore, it is crucial to develop new mechanisms for FGL that should consider the unique characteristics of GNNs to achieve an optimal trade-off between privacy protection and model utility. Furthermore, the convergence analysis of differentially private FGL algorithms is still an open research problem. A tight convergence upper bound not only theoretically assures rapid convergence, but also facilitates an empirical examination of how various hyperparameters influence convergence rates. Such insights are pivotal for fine-tuning parameters or the development of innovative optimization strategies [53]. Existing analyses, such as those presented in [54, 89], do not consider GNN-specific processes like propagation and aggregation. Consequently, there is an urgent need for advanced convergence analysis approaches suitable for DP-FGL. Another important research direction is to evaluate the performance of differentially private FGL algorithms against malicious attacks, such as inference attacks [59, 66] and reconstruction attacks [23, 114]. This research will contribute significantly to the robustness and reliability of DP-based FL.

A.9 Federated Heterogeneous Graph Learning

Heterogeneous graphs (HGs), which are characterized by multiple types of nodes and relations, are widely encountered in real-world scenarios, including social networks and recommendation systems. These graphs offer a more comprehensive representation of complex systems, encompassing diverse information and richer semantics compared to homogeneous graphs, making them particularly valuable for modeling and analysis in various practical applications.

Therefore, HGs in federated settings present more complex scenarios and pose greater challenges for federated learning. Beyond the typical feature heterogeneity, label heterogeneity, and structural heterogeneity encountered in homogeneous graphs within FGL, the diverse relationships inherent in HGs introduce an additional layer of complexity—relation heterogeneity among different clients [94]. This diversity in relational types across clients complicates the learning process, as it requires handling variations in how different clients structure and interpret these relationships. Moreover, since meta-paths within HGs carry specific semantic information, differences in relation types lead to semantic heterogeneity among clients, further complicating the design of effective FGL algorithms [96]. Consequently, existing federated heterogeneous graph learning (FHGL) methods are often highly tailored to specific scenarios and needs, leading to a lack of standardized approaches for these complex learning environments. This variability underscores the need for more generalized and adaptable FHGL frameworks that can effectively manage the diverse challenges posed by HGs in federated settings.

In the following subsection, we provide a comprehensive summary of 6 FHGL models, categorizing them into two setups based on their application scenarios: *Relation Type Sharing* and *Relation Type Protection*. For each algorithm, we outline the core ideas and detail their FL strategies, encoders, and client partitioning approaches, as presented in Table 11. Additionally, we introduce a basic FHGL framework within OpenFGL to assist users in efficiently conducting FL experiments on heterogeneous graphs, thereby facilitating more streamlined research and development in this area.

A.9.1 Relation Type Sharing. This FHGL setup typically defines global relation types on the server side, with each client constructing its local heterogeneous graph according to these predefined standards. The focus is primarily on addressing the challenges associated with federated training instability and performance degradation that arise due to relation heterogeneity across clients.

As one of the pioneering FHGL models, FDRS [46] addresses the cold start problem caused by sparse data within individual clients. FDRS utilizes a two-level aggregation heterogeneous graph convolutional network (HGCN) within each client. This approach facilitates message passing between nodes of the same type through object-level aggregation, while type-level aggregation updates information across different types of nodes. After local training, each client uploads its model parameters to the server, where the FedAvg [63] algorithm aggregates and updates the parameters from all clients. Although FDRS does not implement a specific communication strategy tailored to FHGL, it effectively demonstrates the benefits of cross-client HGNNs. Moreover, it underscores the necessity of learning heterogeneous relations from other clients to enhance and enrich the local information available to each client, thereby improving overall model performance.

FedAHE [84] underscores the significance of recognizing the diversity of meta-path instances in HGs, stressing that this diversity should be accounted for not only within individual clients but also across different clients. To address the semantic heterogeneity that arises among clients due to differences in meta-paths, the authors propose dynamic weighted aggregation of parameters (FedDWA). This mechanism aims to harmonize the variations in meta-paths, thereby reducing semantic discrepancies across clients. During the training process, after each round of aggregation on the server, FedAHE evaluates whether the model version gap between clients exceeds a predefined threshold. If the version gap surpasses this limit, the server initiates a synchronization process by distributing the latest model weights to all clients, ensuring that local models are updated and aligned with the most recent global model. This approach helps to maintain consistency across the federation, thereby

Table 11: Summary of federated heterogeneous graph learning algorithms. Asterisk (*) indicates that the model has been improved, and "None" indicates that it is not described in the original paper.

Client Setup	Method	Basic FL	HGNN Encoder	Datasets	Graph Partitioning
Relation Type Sharing	**FDRS** [46]	FedAvg [63]	HGCN*	Epinions	None
	FedAHE [84]	FedDWA	HAN [87]	ACM DBLP Aminer	None
	FedDA [26]	Dynamic Activation	Simple-HGN [60]	DBLP Amazon LastFM PubMed	Dominant
	FedHGNN [96]	FedAvg	HAN	ACM DBLP Yelp Douban	Ego Graph
Relation Type Protection	**FedHGN** [17]	FedAvg*	RGCN* [74]	AIFB MUTAG BGS	Random Edges Random Edge Types
	FedLIT [94]	Dynamic Clustering	RGCN*	DBLP PubMed NELL MIMIC3	Distinct Dominant Balanced

enhancing the overall robustness and effectiveness of the FL process in the presence of meta-path diversity.

Compared to FGL algorithms, FedDA [26] analyzes the unique characteristics of FL on HGs: that is, only a small amount of client parameters need to be uploaded in each communication round to achieve rapid convergence. Based on these findings, FedDA proposes a dynamic activation strategy, which achieves efficient training by dynamically selecting a subset of clients for each round of aggregation. Furthermore, considering the complex relations in HGs, FedDA introduces the D-HGN model, which decouples the parameters of relation types to allow partial updates of relation parameters on the server side instead of the entire model parameters.

FedHGNN [96] identifies the semantic broken issue that may arise due to the incompleteness of HGs across different clients. To mitigate this issue, FedHGNN proposes a semantic-preserving user interaction publishing algorithm, which captures cross-client semantic information by uploading a shared pattern to the server side. Furthermore, to prevent the shared pattern from leaking the local client's privacy, FedHGNN introduces a two-stage perturbation mechanism to disturb the interactions within the local client and theoretically verifies that this strategy satisfies both semantic privacy and interaction privacy guided by semantics. This strategy only requires one communication before the FL training and then utilizes HAN [87] for encoding within the client and FedAvg for cross-client communication training.

A.9.2 Relation Type Protection. This FHGL setup assumes that the relation type of each client is private and protected, and the server cannot know the specific relation types of clients. Therefore, related models usually adopt some heuristic manners to achieve personalized aggregation and updating of local model parameters without exposing the specific client-side edge relations.

FedHGN [17] highlights that the complex relations in HGs make it challenging for clients to collect and maintain all types of relations. To address this issue, they propose a schema-weight decoupling strategy, which involves performing basis decomposition on the weight matrix of locally specific relations to form relation-specific coefficients β and globally shared basis decomposition. Meanwhile, FedHGN updates β by heuristically matching the minimum distance between relation-specific coefficients of different clients.

FedLIT [94] is primarily designed for vertical FGL. For example, in the same city, different institutions may have similar user samples, but due to institutional differences (such as hospitals and shopping malls), the relations between users are also different. To address this, FedLIT proposes a dynamic latent link-type-aware clustered strategy. This strategy clusters within clients based on edge-type embeddings to obtain local centroids for each relation, and then performs a secondary clustering of the local centroids on the server side to obtain global centroids. Similarly, FedLIT employs heuristic methods to group each local centroid on the server side and aggregates and updates specific relation projection matrice.

A.9.3 Basic FHGL on OpenFGL. Due to the inherent complexity of HGs, the aforementioned algorithms are designed only for specific scenarios rather than a general federated graph scenario. To facilitate users in quickly conducting FGL tasks on HG datasets, we provide a basic FHGL model on OpenFGL for users to perform simulation experiments. Specifically, OpenFGL offers two types of heterogeneous graph partitioning strategies: (1) **Relation Type Sharing**: This strategy follows the traditional FL setup, where

Table 12: Statistics of heterogeneous graph datasets.

Dataset	#Node	#Edge	#Node Type		#Relation		#Classes
ACM	10,942	547,872	# Author (A) # Paper (P)	# Term (T) # Subject (S)	P⇌ A P⇌ P	P⇌ T P⇌ S	3
DBLP4HGB	26,128	239,566	# Author (A) # Paper (P)	# Term (T) # Venue (V)	A⇌ P P⇌ V	P⇌ T	4
DBLP4MGN	26,128	296,563	# Author (A) # Paper (P)	# Term (T) # Conference (C)	A⇌ P P⇌ C	P⇌ T	4
IMDB	21,420	86,624	# Movie (M) # Actor (A)	# Director (D)	M⇌ D	M⇌ A	3

Table 13: The performances (%) of federated heterogeneous graph learning on node classification.

Methods		Datasets			
Basic FL	HGNN Encoder	ACM	DBLP4HGB	DBLP4MGN	IMDB
FedAvg [63]	HAN	86.36±1.40	78.32±4.17	77.19±4.26	50.39±0.98
	RGCN	89.74±0.24	80.86±0.59	80.39±0.47	60.42±0.42
	HGT	84.25±1.30	79.29±0.78	78.28±1.53	55.86±0.73
FedDC [20]	HAN	83.81±1.03	72.55±4.96	73.62±4.62	52.31±0.95
	RGCN	71.53±4.45	81.23±0.57	81.27±0.47	60.95±0.62
	HGT	85.22±2.16	79.61±0.78	78.28±1.53	50.57±0.91
Moon [47]	HAN	86.03±1.46	77.02±3.91	77.19±3.99	51.30±1.11
	RGCN	89.50±0.51	81.24±0.42	81.42±0.38	60.99±0.72
	HGT	82.25±2.53	77.75±1.36	79.42±1.26	56.80±1.11
Scaffold [37]	HAN	86.66±1.62	77.66±4.18	78.02±4.12	55.54±0.60
	RGCN	89.87±0.32	81.07±0.65	80.86±0.76	59.02±0.07
	HGT	87.78±0.91	79.24±0.83	79.46±0.71	55.96±0.75

nodes of the target type are partitioned among multiple clients according to a Dirichlet distribution, and other types of nodes are extracted based on their relations with the target nodes. In this way, the relations and node types in each client are the same, and these types are considered globally shared. (2) **Relation Type Protection**: According to the Random Edge Types strategy provided by FedHGN [17], different types of relations are randomly partitioned among different clients. This method requires that the relation types in each client cannot be shared, necessitating the establishment of specific FL strategies to protect this privacy.

For heterogeneous graph datasets, we used DBLP4HGB and ACM datasets provided by Simple-HGN [60], as well as the DBLP4MGN and IMDB datasets provided by MAGNN [18]. The statistical information of the datasets is shown in Table 12. Here, we provide some experimental results, as shown in Table 13. The heterogeneous subgraph partitioning strategy follows the relation type sharing strategy and is divided into 10 clients. All experiments are repeated ten times, and the mean and standard deviation are reported.

Although in FHGL, different HGNNs [33, 74, 87] and FL methods [20, 37, 47, 63] can be combined to achieve the basic framework, it lacks distributed characteristics. For example, in centralized learning, the attention mechanism is considered an effective method of identifying relations [60]. However, in Table 13, HAN and HGT based on different attention mechanisms perform worse in the FHGL scenario compared to the RGCN. We speculate that this is because the attention mechanism requires additional parameters, making it more prone to over-fitting and stronger local biases. Additionally, the information loss caused by meta-paths is further amplified in FGL, resulting in HAN achieving the worst performance. It is worth mentioning that in most cases, the performance of the same HGNN model shows almost no significant differences across different FL methods. Therefore, for distributed HGs, designing more reasonable HGNNs seems to be a more effective approach. OpenFGL has user-friendly extensibility, allowing users to quickly experiment with their own HGNN and FHGL strategies. Given the practical value of FL and the more realistic modeling scenarios of HG, OpenFGL will inspire more users to conduct research on FHGL.

In the future, we will continue to enhance the adaptability of OpenFGL on HGs, such as incorporating more heterogeneous graph datasets with diverse scenarios (such as PubMed [94], Freebase [60], and OGB-MAG [32]), developing more advanced and efficient heterogeneous GNNs [100], expanding downstream tasks (link prediction, graph classification [99]), and implementing more realistic distributed partitioning strategies [17].

REFERENCES

[1] 2024. OpenFGL Technical Report. In https://github.com/xkLi-Allen/OpenFGL.
[2] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In Proceedings of ACM SIGSAC Conference on Computer and Communications Security, CCS.
[3] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD.
[4] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 2016. 3d semantic parsing of large-scale indoor spaces. In Proceedings of the IEEE conference on computer vision and pattern recognition. 1534–1543.
[5] Jinheon Baek, Wonyong Jeong, Jiongdao Jin, Jaehong Yoon, and Sung Ju Hwang. 2023. Personalized Subgraph Federated Learning. (2023).
[6] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment 2008, 10 (2008), P10008.
[7] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. 2005. Protein function prediction via graph kernels. Bioinformatics 21, suppl_1 (2005), i47–i56.
[8] Chuan Chen, Ziyue Xu, Weibo Hu, Zibin Zheng, and Jie Zhang. 2024. Fedgl: Federated graph learning framework with global self-supervision. Information Sciences 657 (2024), 119976.
[9] Ming Chen, Zhewei Wei, Bolin Ding, Yaliang Li, Ye Yuan, Xiaoyong Du, and Ji-Rong Wen. 2020. Scalable graph neural networks via bidirectional propagation. Advances in Neural Information Processing Systems, NeurIPS (2020).
[10] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In International Conference on Machine Learning, ICML.
[11] Yutong Dai, Zeyuan Chen, Junnan Li, Shelby Heinecke, Lichao Sun, and Ran Xu. 2023. Tackling data heterogeneity in federated learning with class prototypes. In Proceedings of the AAAI Conference on Artificial Intelligence, AAAI.
[12] Ameya Daigavane, Gagan Madan, Aditya Sinha, Abhradeep Guha Thakurta, Gaurav Aggarwal, and Prateek Jain. 2021. Node-level differentially private graph neural networks. In arXiv preprint arXiv:2111.15521.
[13] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. Journal of medicinal chemistry 34, 2 (1991), 786–797.
[14] Frederik Diehl, Thomas Brunner, Michael Truong Le, and Alois Knoll. 2019. Towards graph pooling by edge contraction. In ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data.
[15] Paul D Dobson and Andrew J Doig. 2003. Distinguishing enzyme structures from non-enzymes without alignments. Journal of molecular biology 330, 4 (2003), 771–783.
[16] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In Theory of Cryptography: Third Theory of Cryptography Conference, TCC. Springer.
[17] Xinyu Fu and Irwin King. 2023. FedHGN: A Federated Framework for Heterogeneous Graph Neural Networks. In Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI.
[18] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In Proceedings of the ACM Web Conference, WWW.
[19] Hongyang Gao and Shuiwang Ji. 2019. Graph u-nets. In International Conference on Machine Learning, ICML.
[20] Liang Gao, Huazhu Fu, Li Li, Yingwen Chen, Ming Xu, and Cheng-Zhong Xu. 2022. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR.
[21] Zihao Gao, Huifang Ma, Xiaohui Zhang, Yike Wang, and Zheyu Wu. 2023. Similarity measures-based graph co-contrastive learning for drug–disease association prediction. Bioinformatics 39, 6 (2023), btad357.
[22] Anna Gaulton, Anne Hersey, Michał Nowotka, A Patricia Bento, Jon Chambers, David Mendez, Prudence Mutowo, Francis Atkinson, Louisa J Bellis, Elena Cibrián-Uhalte, et al. 2017. The ChEMBL database in 2017. Nucleic acids research 45, D1 (2017), D945–D954.
[23] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting gradients-how easy is it to break privacy in federated learning?. In Advances in Neural Information Processing Systems, NeurIPS.
[24] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. arXiv preprint arXiv:1802.06893 (2018).
[25] Zishan Gu, Ke Zhang, Guangji Bai, Liang Chen, Liang Zhao, and Carl Yang. 2023. Dynamic activation of clients and parameters for federated learning over heterogeneous graphs. In International Conference on Data Engineering, ICDE. IEEE.
[26] Zishan Gu, Ke Zhang, Guangji Bai, Liang Chen, Liang Zhao, and Carl Yang. 2023. Dynamic Activation of Clients and Parameters for Federated Learning over Heterogeneous Graphs. In International Conference on Data Engineering, ICDE.
[27] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. 2018. Pcpnet learning local shape properties from raw point clouds. In Computer graphics forum, Vol. 37. Wiley Online Library, 75–85.
[28] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. Advances in Neural Information Processing Systems, NeurIPS (2017).
[29] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems, NeurIPS.
[30] Chaoyang He, Keshav Balasubramanian, Emir Ceyani, Carl Yang, Han Xie, Lichao Sun, Lifang He, Liangwei Yang, S Yu Philip, Yu Rong, et al. 2021. FedGraphNN: A Federated Learning Benchmark System for Graph Neural Networks. In International Conference on Learning Representations, ICLR Workshop on Distributed and Private Machine Learning.
[31] Christoph Helma, Ross D. King, Stefan Kramer, and Ashwin Srinivasan. 2001. The predictive toxicology challenge 2000–2001. Bioinformatics 17, 1 (2001), 107–108.
[32] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. Advances in Neural Information Processing Systems, NeurIPS (2020).
[33] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In Proceedings of the ACM Web Conference, WWW.
[34] Wenke Huang, Guancheng Wan, Mang Ye, and Bo Du. 2023. Federated graph semantic and structural learning. In Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI.
[35] Woochang Hyun, Jaehong Lee, and Bongwon Suh. 2023. Anti-Money Laundering in Cryptocurrency via Multi-Relational Graph Neural Network. In Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, 118–130.
[36] Zhida Jiang, Yang Xu, Hongli Xu, Zhiyuan Wang, and Chunming Qiao. 2023. Clients Help Clients: Alternating Collaboration for Semi-Supervised Federated Learning. In International Conference on Data Engineering, ICDE. IEEE.
[37] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In International Conference on Machine Learning, ICML.
[38] George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing 20, 1 (1998), 359–392.
[39] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In International Conference on Learning Representations, ICLR.
[40] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations, ICLR.
[41] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-attention graph pooling. In International Conference on Machine Learning, ICML.
[42] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD.
[43] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford large network dataset collection. (2014).
[44] Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, et al. 2021. Datasets: A community library for natural language processing. arXiv preprint arXiv:2109.02846 (2021).
[45] 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. Proceedings of the ACM on Management of Data, SIGMOD (2024).
[46] Nianzhe Li, Hanwen Liu, Shunmei Meng, and Qianmu Li. 2023. FDRS: Federated Diversified Recommender System Based on Heterogeneous Graph Convolutional Network. In International Conferences on High Performance Computing and Communications, HPCC.
[47] Qinbin Li, Bingsheng He, and Dawn Song. 2021. Model-contrastive federated learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR.
[48] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. Proceedings of Machine Learning and Systems, MLSys (2020).

[49] Xunkai Li, Ronghui Guo, Jianwen Chen, Youpeng Hu, Meixia Qu, and Bin Jiang. 2022. Effective Hybrid Graph and Hypergraph Convolution Network for Collaborative Filtering. Neural Comput. Appl. 35, 3 (sep 2022), 2633–2646.

[50] Xunkai Li, Meihao Liao, Zhengyu Wu, Daohan Su, Wentao Zhang, Rong-Hua Li, and Guoren Wang. 2024. LightDiC: A Simple Yet Effective Approach for Large-Scale Digraph Representation Learning. Proceedings of the VLDB Endowment (2024).

[51] Xunkai Li, Zhengyu Wu, Wentao Zhang, Henan Sun, Rong-Hua Li, and Guoren Wang. 2024. AdaFGL: A New Paradigm for Federated Node Classification with Topology Heterogeneity. arXiv preprint arXiv:2401.11750 (2024).

[52] Xunkai Li, Zhengyu Wu, Wentao Zhang, Yinlin Zhu, Rong-Hua Li, and Guoren Wang. 2023. FedGTA: Topology-Aware Averaging for Federated Graph Learning. Proceedings of the VLDB Endowment (2023).

[53] Yipeng Li and Xinchen Lyu. 2024. Convergence Analysis of Sequential Federated Learning on Heterogeneous Data. In Advances in Neural Information Processing Systems, NeurIPS.

[54] Zhize Li, Haoyu Zhao, Boyue Li, and Yuejie Chi. 2022. SoteriaFL: A unified framework for private federated learning with communication compression. In Advances in Neural Information Processing Systems, NeurIPS.

[55] Ling Liang, Jilan Lin, Zheng Qu, Ishtiyaque Ahmad, Fengbin Tu, Trinabh Gupta, Yufei Ding, and Yuan Xie. 2023. Spg: Structure-private graph database via squeezepir. Proceedings of the VLDB Endowment (2023).

[56] Daniil Likhobaba, Nikita Pavlichenko, and Dmitry Ustalov. 2023. Toloker Graph: Interaction of Crowd Annotators. (2023). https://doi.org/10.5281/zenodo.7620795

[57] D. Lim, F. Hohne, X. Li, S. L. Huang, V. Gupta, O. Bhalerao, and S. N. Lim. 2021. Large Scale Learning on Non-Homophilous Graphs: New Benchmarks and Strong Simple Methods. In Advances in Neural Information Processing Systems, NeurIPS.

[58] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. 2022. Revisiting heterophily for graph neural networks. Advances in Neural Information Processing Systems, NeurIPS (2022).

[59] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. 2021. Feature inference attack on model predictions in vertical federated learning. In International Conference on Data Engineering, ICDE.

[60] Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. 2021. Are we really making much progress?: Revisiting, benchmarking and refining heterogeneous graph neural networks. In Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD.

[61] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. 2021. Is homophily a necessity for graph neural networks? International Conference on Learning Representations, ICLR (2021).

[62] Zheng Ma, Junyu Xuan, Yu Guang Wang, Ming Li, and Pietro Liò. 2020. Path integral based convolution and pooling for graph neural networks. Advances in Neural Information Processing Systems, NeurIPS (2020).

[63] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. Artificial Intelligence and Statistics (2017).

[64] Ilya Mironov. 2017. Rényi differential privacy. In IEEE Computer Security Foundations Symposium, CSF.

[65] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In Proceedings of the IEEE conference on computer vision and pattern recognition. 5115–5124.

[66] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In IEEE Symposium on Security and Privacy, SP.

[67] Qiying Pan, Yifei Zhu, and Lingyang Chu. 2023. Lumos: Heterogeneity-aware federated graph learning over decentralized devices. In International Conference on Data Engineering, ICDE. IEEE.

[68] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. In International Conference on Learning Representations, ICLR.

[69] Oleg Platonov, Denis Kuznedelev, Artem Babenko, and Liudmila Prokhorenkova. 2023. Characterizing graph datasets for node classification: Beyond homophily-heterophily dichotomy. Advances in Neural Information Processing Systems, NeurIPS (2023).

[70] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. 2023. A critical look at the evaluation of GNNs under heterophily: are we really making progress? International Conference on Learning Representations, ICLR (2023).

[71] Yuxin Qiu. 2023. Default Risk Assessment of Internet Financial Enterprises Based on Graph Neural Network. In IEEE Information TechnoLoGy, Networking, Electronic and Automation Control Conference, Vol. 6. IEEE, 592–596.

[72] Zongshuai Qu, Tao Yao, Xinghui Liu, and Gang Wang. 2023. A Graph Convolutional Network Based on Univariate Neurodegeneration Biomarker for Alzheimer's Disease Diagnosis. IEEE Journal of Translational Engineering in Health and Medicine (2023).

[73] Kaspar Riesen and Horst Bunke. 2008. IAM graph database repository for graph based pattern recognition and machine learning. In Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, SSPR & SPR 2008, Orlando, USA, December 4-6, 2008. Proceedings. Springer, 287–297.

[74] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In European Semantic Web Conference, ESWC.

[75] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811.05868 (2018).

[76] Kuangchi Sun and Aijun Yin. 2024. Multisensor Graph Adaptive Federated Generalization for Helicopter Transmission System Fault Diagnosis. IEEE Transactions on Instrumentation and Measurement 73 (2024), 1–11.

[77] Lichao Sun and Lingjuan Lyu. 2021. Federated model distillation with noise-free differential privacy. In Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI.

[78] Jeffrey J Sutherland, Lee A O'brien, and Donald F Weaver. 2003. Spline-fitting with a genetic algorithm: A method for developing classification structure-activity relationships. Journal of chemical information and computer sciences 43, 6 (2003), 1906–1915.

[79] Yue Tan, Yixin Liu, Guodong Long, Jing Jiang, Qinghua Lu, and Chengqi Zhang. 2023. Federated learning on non-iid graphs via structural knowledge sharing. In Proceedings of the AAAI Conference on Artificial Intelligence, AAAI.

[80] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. 2022. Fedproto: Federated prototype learning across heterogeneous clients. In Proceedings of the AAAI Conference on Artificial Intelligence, AAAI.

[81] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In International Conference on Learning Representations, ICLR.

[82] Nikil Wale, Ian A Watson, and George Karypis. 2008. Comparison of descriptor spaces for chemical compound retrieval and classification. Knowledge and Information Systems 14 (2008), 347–375.

[83] Guancheng Wan, Wenke Huang, and Mang Ye. 2024. Federated Graph Learning under Domain Shift with Generalizable Prototypes. In Proceedings of the AAAI Conference on Artificial Intelligence, AAAI.

[84] Junfu Wang, Yawen Li, Meiyu Liang, and Ang Li. 2022. Embedding Representation of Academic Heterogeneous Information Networks Based on Federated Learning. In IEEE International Conference on Cloud Computing and Intelligent Systems, CCIS.

[85] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. 2020. Microsoft academic graph: When experts are not enough. Quantitative Science Studies 1, 1 (2020), 396–413.

[86] Qinyong Wang, Hongzhi Yin, Tong Chen, Junliang Yu, Alexander Zhou, and Xiangliang Zhang. 2022. Fast-adapting and privacy-preserving federated recommender system. The VLDB Journal 31, 5 (2022), 877–896.

[87] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In Proceedings of the ACM Web Conference, WWW.

[88] Zhen Wang, Weirui Kuang, Yuexiang Xie, Liuyi Yao, Yaliang Li, Bolin Ding, and Jingren Zhou. 2022. FederatedScope-GNN: Towards a Unified, Comprehensive and Efficient Package for Federated Graph Learning. Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD (2022).

[89] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. IEEE Transactions on Information Forensics and Security, TIFS 15 (2020), 3454–3469.

[90] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In International Conference on Machine Learning, ICML.

[91] Honghu Wu, Xiangrong Zhu, and Wei Hu. 2024. A Blockchain System for Clustered Federated Learning with Peer-to-Peer Knowledge Transfer. Proceedings of the VLDB Endowment (2024).

[92] Yuncheng Wu, Naili Xing, Gang Chen, Tien Tuan Anh Dinh, Zhaojing Luo, Beng Chin Ooi, Xiaokui Xiao, and Meihui Zhang. 2023. Falcon: A privacy-preserving and interpretable vertical federated learning system. Proceedings of the VLDB Endowment (2023).

[93] Han Xie, Jing Ma, Li Xiong, and Carl Yang. 2021. Federated graph classification over non-iid graphs. Advances in Neural Information Processing Systems, NeurIPS (2021).

[94] Han Xie, Li Xiong, and Carl Yang. 2023. Federated Node Classification over Graphs with Latent Link-type Heterogeneity. In Proceedings of the ACM Web Conference, WWW.

[95] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? (2019).
[96] Bo Yan, Yang Cao, Haoyu Wang, Wenchuan Yang, Junping Du, and Chuan Shi. 2024. Federated Heterogeneous Graph Neural Network for Privacy preserving Recommendation. In Proceedings of the ACM Web Conference, WWW.
[97] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD.
[98] Liangwei Yang, Shengjie Wang, Yunzhe Tao, Jiankai Sun, Xiaolong Liu, Philip S Yu, and Taiqing Wang. 2023. DGRec: Graph Neural Network for Recommendation with Diversified Embedding Generation. In Proceedings of the ACM International Conference on Web Search and Data Mining, WSDM.
[99] Qiang Yang, Changsheng Ma, Qiannan Zhang, Xin Gao, Chuxu Zhang, and Xiangliang Zhang. 2023. Counterfactual Learning on Heterogeneous Graphs with Greedy Perturbation. In Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD.
[100] Xiaocheng Yang, Mingyu Yan, Shirui Pan, Xiaochun Ye, and Dongrui Fan. 2023. Simple and Efficient Heterogeneous Graph Neural Network. In Proceedings of the Association for the Advancement of Artificial Intelligence, AAAI.
[101] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In International Conference on Machine Learning, ICML.
[102] Feng Yao, Qian Tao, Wenyuan Yu, Yanfeng Zhang, Shufeng Gong, Qiange Wang, Ge Yu, and Jingren Zhou. 2023. RAGraph: A Region-Aware Framework for Geo-Distributed Graph Processing. Proceedings of the VLDB Endowment (2023).
[103] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. 2016. A scalable active framework for region annotation in 3d shape collections. ACM Transactions on Graphics (ToG) 35, 6 (2016), 1–12.
[104] Wei Yuan, Liang Qu, Lizhen Cui, Yongxin Tong, Xiaofang Zhou, and Hongzhi Yin. 2024. Hetefedrec: Federated recommender systems with model heterogeneity. In International Conference on Data Engineering, ICDE. IEEE.
[105] Xiaoming Yuan, Jiahui Chen, Jiayu Yang, Ning Zhang, Tingting Yang, Tao Han, and Amir Taherkordi. 2022. Fedstn: Graph representation driven federated learning for edge computing enabled urban traffic flow prediction. IEEE Transactions on Intelligent Transportation Systems 24, 8 (2022), 8738–8748.
[106] Ye Yuan, Delong Ma, Zhenyu Wen, Zhiwei Zhang, and Guoren Wang. 2021. Subgraph matching over graph federation. Proceedings of the VLDB Endowment (2021).
[107] Jianqing Zhang, Yang Liu, Yang Hua, and Jian Cao. 2024. FedTGP: Trainable Global Prototypes with Adaptive-Margin-Enhanced Contrastive Learning for Data and Model Heterogeneity in Federated Learning. In Proceedings of the AAAI Conference on Artificial Intelligence, AAAI.
[108] Ke Zhang, Lichao Sun, Bolin Ding, Siu Ming Yiu, and Carl Yang. 2024. Deep efficient private neighbor generation for subgraph federated learning. In Proceedings of SIAM International Conference on Data Mining, SDM.
[109] Ke Zhang, Lichao Sun, Bolin Ding, Siu Ming Yiu, and Carl Yang. 2024. Deep efficient private neighbor generation for subgraph federated learning. In SIAM International Conference on Data Mining, SDM.
[110] 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, NeurIPS (2021).
[111] Wentao Zhang, Ziqi Yin, Zeang Sheng, Yang Li, Wen Ouyang, Xiaosen Li, Yangyu Tao, Zhi Yang, and Bin Cui. 2022. Graph Attention Multi-Layer Perceptron. Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD (2022).
[112] Xinyi Zhang, Qichen Wang, Cheng Xu, Yun Peng, and Jianliang Xu. 2024. FedKNN: Secure Federated k-Nearest Neighbor Search. Proceedings of the ACM on Management of Data, SIGMOD (2024).
[113] Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S Yu. 2022. Graph neural networks for graphs with heterophily: A survey. arXiv preprint arXiv:2202.07082 (2022).
[114] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. In Advances in Neural Information Processing Systems, NeurIPS.
[115] Yinlin Zhu, Xunkai Li, Zhengyu Wu, Di Wu, Miao Hu, and Rong-Hua Li. 2024. FedTAD: Topology-aware Data-free Knowledge Distillation for Subgraph Federated Learning. arXiv preprint arXiv:2404.14061 (2024).
[116] Daniel Zügner and Stephan Günnemann. 2019. Adversarial Attacks on Graph Neural Networks via Meta Learning. In International Conference on Learning Representations, ICLR.