Conformal Prediction for Federated Graph Neural Networks with Missing Neighbor Information

Ömer Faruk Akgül[1] **Rajgopal Kannan**[2] **Viktor Prasanna**[1]

[1]University of Southern California, Los Angeles
[2]DEVCOM ARL Army Research Office

Abstract

Uncertainty quantification is essential for reliable federated graph learning, yet existing methods struggle with decentralized and heterogeneous data. In this work, we first extend Conformal Prediction (CP), a well-established method for uncertainty quantification, to federated graph learning, formalizing conditions for CP validity under partial exchangeability across distributed subgraphs. We prove that our approach maintains rigorous coverage guarantees even with client-specific data distributions. Building on this foundation, we address a key challenge in federated graph learning: missing neighbor information, which inflates CP set sizes and reduces efficiency. To mitigate this, we propose a variational autoencoder (VAE)-based architecture that reconstructs missing neighbors while preserving data privacy. Empirical evaluations on real-world datasets demonstrate the effectiveness of our method: our theoretically grounded federated training strategy reduces CP set sizes by 15.4%, with the VAE-based reconstruction providing an additional 4.9% improvement, all while maintaining rigorous coverage guarantees.

1 INTRODUCTION

Graph Neural Networks (GNNs) have significantly advanced graph data mining, demonstrating strong performance across various domains, including social platforms, e-commerce, transportation, bioinformatics, and healthcare [Hamilton et al., 2018, Kipf and Welling, 2017, Wu et al., 2022, Zhang et al., 2021b]. In many real-world scenarios, graph data is inherently distributed due to the nature of data generation and collection processes [Zhou et al., 2020]. For example, data from social networks, healthcare systems, and financial institutions [Liu et al., 2019] is often generated by multiple independent entities, leading to fragmented and distributed graph structures. This distributed nature of graph data poses unique challenges when training GNNs, such as the need to address data privacy, ownership, and regulatory constraints [Zhang et al., 2021a].

Federated Learning (FL) emerges as a solution, allowing collaborative model training without centralized data sharing [McMahan et al., 2017, Kairouz et al., 2021]. FL addresses data isolation issues and has been widely used in various applications, including computer vision and natural language processing [Li et al., 2020]. However, applying FL to graph data introduces unique challenges, such as incomplete node neighborhoods and missing links across distributed subgraphs [Zhang et al., 2021a]. These missing connections can degrade model performance and increase uncertainty, underscoring the need for robust uncertainty quantification techniques.

Conformal Prediction [Vovk et al., 2005] offers a promising framework for producing statistically guaranteed uncertainty estimates, providing user-specified confidence levels to construct prediction sets with provable coverage guarantees. Specifically, with a miscoverage level $\alpha \in (0, 1)$, CP uses calibration data to generate prediction sets for new instances, ensuring the true outcome is contained within them with probability at least $1 - \alpha$.

While CP has been explored in natural language processing [Kumar et al., 2023], computer vision [Angelopoulos et al., 2020], federated learning [Lu et al., 2023], and GNNs [Zargarbashi et al., 2023, Huang et al., 2024], its application in federated graph learning remains underexplored. A primary challenge is ensuring the *exchangeability* assumption, critical for CP's validity, holds in partitioned graph data, which may not be the case due to data heterogeneity across clients.

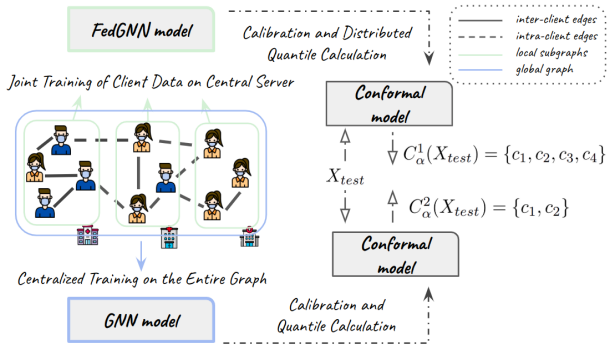

Figure 1: **Overview of federated conformal prediction for graph-structured data.** A simplified scenario involving patient data distributed across three hospitals, highlighting both intra-client (solid lines) and inter-client (dashed lines) connections. In federated settings, inter-client links are often missing, despite their real-world presence, leading to fragmented subgraphs. The FedGNN model optimizes a global model through local updates on each client, while the centralized GNN model operates on the complete graph with all connections intact, serving as a performance benchmark. Missing inter-client links result in larger conformal prediction sets, as shown by $C^1_\alpha(X_{\text{test}})$ (prediction set from FedGNN) and $C^2_\alpha(X_{\text{test}})$ (prediction set from centralized GNN), illustrating how missing links affect model uncertainty.

In this paper, we investigate conformal prediction within a federated graph learning framework, where multiple clients, each with distinct local data distributions P_k over node-feature-label pairs (x, y), collaboratively train a shared global model while experiencing missing neighbor information. Our objective is to construct prediction sets with marginal coverage guarantees for unseen data drawn from a global test distribution $Q_{\text{test}} = \sum_{k=1}^{K} p_k P_k$, where p_k denotes the mixing weight for client k. However, heterogeneity across the client distributions P_k can violate the exchangeability assumption required by conformal prediction, undermining the validity of coverage guarantees and leading to larger, less informative prediction sets [Huang et al., 2024]. This issue is further compounded by the absence of inter-client links, which limits structural context and increases uncertainty due to incomplete neighborhood information, as illustrated in Figure 1.

We extend the theoretical framework of *partial exchangeability* to graphs within the federated learning setting, addressing the challenges posed by data heterogeneity across the client subgraphs. Our analysis reveals inefficiencies in the size of the conformal prediction sets attributable to missing links. To counteract these inefficiencies, we introduce a novel framework designed to generate missing links across clients, thereby optimizing the size of CP sets.

Our main contributions are summarized as follows:

- We extend Conformal Prediction to federated graph settings, establish the necessary conditions for CP validity and derive theoretical statistical guarantees.
- We analyze how the absence of inter-client links inflates conformal prediction set sizes and propose a method to mitigate this inefficiency through local subgraph completion.
- We demonstrate the effectiveness of our approach through empirical evaluation on four benchmark datasets, showing improved efficiency of CP in federated graph scenarios.

2 RELATED WORK

Recent advancements have applied Conformal Prediction to graph machine learning to enhance uncertainty quantification in GNNs. [Clarkson, 2023] improved calibration and prediction sets for node classification in inductive learning scenarios. [Huang et al., 2024] and [Zargarbashi et al., 2023] focused on reducing prediction set sizes, with the latter enhancing efficiency through the diffusion of node-wise conformity scores and leveraging network homophily.

In Federated Learning, [Lu et al., 2023] extended CP techniques to address data heterogeneity, providing theoretical guarantees for uncertainty quantification in distributed settings. [Zhang et al., 2021a] introduced FedSage and FedSage+, methods for training graph mining models on distributed subgraphs, tackling data heterogeneity and missing links. [Baek et al., 2023] explored personalized weight aggregation based on subgraph similarity in a personalized subgraph FL framework. [Tan et al., 2022] proposed FedProto, which constructs prototypes from local client data to enhance learning across subgraphs, though it does not address privacy concerns related to sharing prototypes.

Despite these efforts, existing work does not fully address the challenges of missing links and subgraph heterogeneity in graph conformal settings. Our work is the first to propose a CP method specifically designed for federated graph learning, addressing both exchangeability violations and inefficiencies caused by missing neighbor information.

3 PRELIMINARIES

We begin by defining the preliminary concepts and notation used throughout this paper. A summary of the key symbols is provided in Table 1 for easy reference.

3.1 CONFORMAL PREDICTION

Conformal prediction is a framework for uncertainty quantification that provides rigorous statistical guarantees. We focus on the split conformal prediction method [Vovk et al.,

Table 1: Summary of key notation.

Symbol	Description
Graphs and Federated Learning	
$\mathcal{G}, \mathcal{V}, \mathcal{E}$	A graph with a set of nodes and edges.
X, Y	Node features and labels.
K	Total number of clients.
$\mathcal{G}^k, \mathcal{V}^k, \mathcal{E}^k$	Subgraph at client k.
P_k	Data distribution at client k.
m_k	Training set size at client k.
n_k	Calibration set size at client k.
Conformal Prediction	
α	Target miscoverage level.
$S(x, y)$	Non-conformity function.
s_v or s_i^k	Non-conformity score for a node.
$\hat{q}_\alpha$	Empirical score quantile (cutoff).
$C_\alpha(x)$	Conformal prediction set.
Generative Model	
c_m	Feature prototype (cluster center).
$\hat{X}$	Aggregated set of all prototype features.
M	Number of feature prototypes per client k.
p	Percentage of new edges to add.

2005], notable for its computational efficiency. The method defines a non-conformity measure $S : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, which quantifies how atypical the true label y is for the input x according to the model's predictions. For classification tasks, $S(x, y)$ might be defined as $1 - f_y(x)$, where $f_y(x)$ is the estimated probability of class y given x.

3.1.1 Quantile Calculation and Prediction Set Construction

Using a calibration dataset $\mathcal{D}_{\text{calib}} = \{(x_i, y_i)\}_{i=1}^n$, we compute the non-conformity scores $S_i = S(x_i, y_i)$ for each calibration example. The cutoff value $\hat{q}_\alpha$ is then determined as the $(1-\alpha)(1+\frac{1}{n})$-th empirical quantile of these scores, i.e., $\hat{q}_\alpha = \text{quantile}\left(\{S_1, \ldots, S_n\}, (1-\alpha)\left(1+\frac{1}{n}\right)\right)$. Given a new input x, the prediction set is constructed as $C_\alpha(x) = \{y \in \mathcal{Y} : S(x, y) \leq \hat{q}\}$. Under the assumption of exchangeability of the data, this method guarantees that the true label y will be contained in $C_\alpha(x)$ with probability at least $1 - \alpha$.

Adaptive Prediction Sets (APS) [Romano et al., 2020] construct prediction sets by accumulating class probabilities. Given a probabilistic classifier that outputs estimated class probabilities $f(x) = (f_1(x), \ldots, f_{|\mathcal{Y}|}(x))$, where $f_j(x)$ is the estimated probability of class j for input x, we sort the classes in descending order to obtain a permutation π such that $f_{\pi(1)}(x) \geq f_{\pi(2)}(x) \geq \ldots \geq f_{\pi(|\mathcal{Y}|)}(x)$. The cumulative probability up to the k-th class is $V(x, k) = \sum_{j=1}^k f_{\pi(j)}(x)$. For each calibration example (x_i, y_i), we compute the non-conformity score $S_i = V(x_i, k_i)$, where k_i is the rank of the true label y_i in the sorted class probabilities for x_i. The cutoff value $\hat{q}$ is then determined as before. The prediction set $C_\alpha(x)$ includes the top k^* classes, where $k^* = \min\{k : V(x, k) \geq \hat{q}\}$ and $C_\alpha(x) = \{\pi(1), \ldots, \pi(k^*)\}$. While we use APS for presenting our main results, other scores like Regularized Adaptive Prediction Sets (RAPS) and Least Ambiguous Set-Valued Classifiers (LAC) are also commonly used. We provide a comparative analysis of these non-conformity scores in Appendix E.

3.1.2 Evaluation Metrics

Our goal is to achieve valid marginal coverage while minimizing the size of the prediction sets. The inefficiency is measured as $\text{Inefficiency}_\alpha = \frac{1}{m}\sum_{j=1}^m |C_\alpha(x_j)|$, on the test set $\mathcal{D}_{\text{test}} = \{(x_j, y_j)\}_{j=1}^m$, where m denotes the number of test samples. Empirical coverage is calculated as $\text{Coverage}_\alpha = \frac{1}{m}\sum_{j=1}^m \mathbb{1}\{y_j \in C_\alpha(x_j)\}$, representing the proportion of test examples where the true label is included in the prediction set.

3.2 GNNS AND FEDERATED GRAPH LEARNING

GNNs effectively capture structural information and node features in graph-structured data [Kipf and Welling, 2017]. Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of n nodes and $\mathcal{E}$ is the set of edges. Each node $v \in \mathcal{V}$ is associated with a feature vector $x_v \in \mathbb{R}^d$, forming the input matrix $X = \{x_v\}_{v \in \mathcal{V}} \in \mathbb{R}^{n \times d}$.

In node classification, the goal is to predict labels for nodes by leveraging both node features and the graph topology. We operate under a transductive learning setting where the full graph $\mathcal{G}$ is available during training and testing, but test labels are withheld. To enable conformal prediction, we partition the node set $\mathcal{V}$ into four disjoint subsets: training, validation, calibration, and test nodes, denoted as $\mathcal{V}_{\text{train}}$, $\mathcal{V}_{\text{valid}}$, $\mathcal{V}_{\text{cal}}$, and $\mathcal{V}_{\text{test}}$, respectively.

A GNN produces node representations through a sequence of message-passing layers. At each layer ℓ, a node u receives messages from its neighbors $v \in \mathcal{N}_u$, computed using a learnable function $\text{MSG}(h_u^{(\ell-1)}, h_v^{(\ell-1)})$, where $h_u^{(\ell-1)}$ denotes the embedding of node u from the previous layer. The incoming messages are aggregated via a permutation-invariant function $\texttt{AGG}$, and the node embedding is updated using a learnable function $\texttt{UPD}$:

$$h_u^{(\ell)} = \texttt{UPD}\Big(\texttt{AGG}\big(\{\texttt{MSG}(h_u^{(\ell-1)}, h_v^{(\ell-1)}) \mid v \in \mathcal{N}_u\}\big), h_u^{(\ell-1)}\Big).$$

The final-layer embeddings are used by a classifier to produce predictions $\mu(X)$, which are used to compute the supervised loss over the training set $\mathcal{V}_{\text{train}}$.

Federated Graph Neural Networks extend GNNs to settings where graph data is distributed across multiple clients [Liu et al., 2024]. A central server coordinates with K clients, each holding a subgraph $\mathcal{G}^k \subset \mathcal{G}$. Each client independently trains a local GNN model on its subgraph using its local labeled nodes. After local training, model parameters θ_k are transmitted to the server, which aggregates them using Federated Averaging (FedAvg) [McMahan et al., 2017]:

$$\theta = \sum_{k=1}^{K} \frac{m_k}{m} \theta_k,$$

where θ_k denotes the parameters of the local GNN model at client k, m_k is the number of local training samples, and $m = \sum_{k=1}^{K} m_k$. The aggregated global model θ is then broadcast back to clients for the next round of training. This process enables collaborative training while preserving data privacy, as no raw data or node features are shared between clients.

In our framework, the FedAvg aggregation is applied to both the GNNs used for node classification and the Variational Graph Autoencoders (VGAEs) used for generating missing neighbor links (Section 5.2). The shared model parameters θ include all learnable weights in the encoder layers, ensuring consistency across clients without exposing private subgraph structure.

3.3 VARIATIONAL AUTOENCODERS

VAEs [Kingma and Welling, 2013] and their extension to graph data, VGAEs [Kipf and Welling, 2016], are fundamental to our approach for generating node features and predicting edges. Both methods utilize deep learning and Bayesian inference to learn latent representations by optimizing the evidence lower bound (ELBO), balancing reconstruction loss and the Kullback-Leibler (KL) divergence between the approximate and prior distributions. The ELBO is defined as: $\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{\text{KL}}[q_\phi(z|x) \| p(z)]$, where $q_\phi(z|x)$ approximates the latent variable z, and $p(z)$ is the prior. In our model, VAEs generate node features, while VGAEs learn latent representations of graph structures for the edge prediction task.

4 CHALLENGES OF CONFORMAL PREDICTION ON FEDERATED GRAPHS

Conformal Prediction on federated graphs faces several challenges that need to be addressed to ensure its applicability and effectiveness in real-world applications. In this section, we elaborate on these challenges. In Section 5, we discuss how we address them.

Exchangeability: A significant challenge in federated graph CP is the violation of the exchangeability principle, which traditional CP methods rely upon [Vovk et al., 2005]. Consider a federated graph learning setting where nodes of the overall graph $\mathcal{V}$ are partitioned into training, validation, calibration, and test sets as $\mathcal{V}_{\text{train}}, \mathcal{V}_{\text{valid}}, \mathcal{V}_{\text{calib}}$, and $\mathcal{V}_{\text{test}}$. These methods presuppose that the distributions of calibration nodes $\mathcal{V}_{\text{calib}}$ and test nodes $\mathcal{V}_{\text{test}}$ are exchangeable during inference, meaning their joint distribution remains unchanged when samples are permuted. This assumption breaks down in federated graph settings for two primary reasons.

First, inherent dependencies among nodes due to their connectivities violate exchangeability if the test data is not present during training. Secondly, the distribution of graph data across different clients in a federated setting tends to vary, leading to non-exchangeable distributions. Specifically, the sets $\mathcal{V}_{\text{calib}}$ and $\mathcal{V}_{\text{test}}$ are not exchangeable, as their respective subsets $\mathcal{V}_{\text{calib}}^{(k)}$ and $\mathcal{V}_{\text{test}}^{(k')}$ may originate from distinct clients ($k \neq k'$). This variability underscores the challenges in assuming uniform data distribution across clients. For example, hospitals specializing in certain medical fields might predominantly treat patients from specific demographic groups, leading to skewed data distributions. Similarly, graph partitioning algorithms like METIS [Karypis and Kumar, 1997], used for simulating subgraph FL scenarios, aim to minimize edge cuts across partitions, often resulting in subgraphs that do not share the same data distribution.

Table 2: Number of partitions (K) and its impact on missing edges (ΔE) and average conformal prediction set sizes ($\Delta|CP|$) across clients. Larger CP set sizes result from both the local training of models and conformal predictors, as well as the increasing number of missing links in client subgraphs.

Dataset	$\lvert E\rvert$	K	ΔE	$\Delta E\%$	$\Delta\lvert CP\rvert\%$
Cora	10,138	5	604	5.96%	34.7%
		10	806	7.95%	43.3%
		20	1,230	12.13%	48.1%
CiteSeer	7,358	5	310	4.21%	54.0%
		10	608	8.26%	57.7%
		20	848	11.52%	62.3%

Missing Neighbor Information: Another significant challenge in federated graph CP is the presence of missing neighbor information across client subgraphs. Consider a scenario where a patient visits multiple hospitals within the same city, maintaining separate records at each location. Due to conflicts of interest, it is impractical for hospitals to share their patient networks, leading to incomplete edge information in the overall graph. In simulations of federated learning based on graph partitioning, increasing the number of clients amplifies the number of missing links between them, as shown in Table 2.

These missing edges, which carry critical neighborhood information, remain uncaptured by any single client subgraph. This absence becomes particularly problematic when CP techniques are applied to partitioned graph data, as it can impair model performance and increase the size of prediction sets due to insufficient coverage of the data's connectivity. Figure 2 illustrates this issue, showing how the increasing number of missing links correlates with larger prediction set sizes through empirical evaluation.

Given these complexities, it is necessary to demonstrate how CP can be applied to non-exchangeable graph data and how the inefficiency caused by missing neighbor information can be mitigated within federated graph environments.

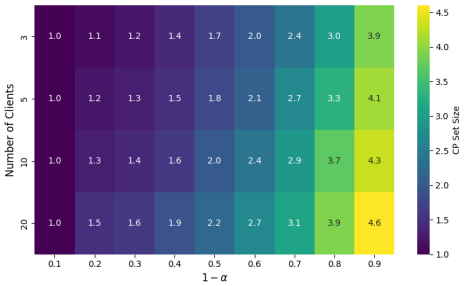

Figure 2: Effect of the number of clients on CP set size for the Cora dataset.

5 METHOD

5.1 PARTIALLY EXCHANGEABLE NON-CONFORMITY SCORES

To deploy Conformal Prediction for federated graph-structured data under a transductive learning setting, we need to ensure the exchangeability condition is met. We adopt the principle of partial exchangeability, as proposed by De Finetti [1980] and applied to non-graph-based models by Lu et al. [2023]. Specifically, we demonstrate that non-conformity scores within each client are permutation invariant when using a permutation-invariant GNN model for training under the transductive setting.

Consider a graph $\mathcal{G}^k = (\mathcal{V}^k, \mathcal{E}^k)$ at client k, where $\mathcal{V}^k$ denotes the set of nodes, $\mathcal{E}^k$ the set of edges, and each node $v \in \mathcal{V}^k$ has a feature vector $x_v \in \mathbb{R}^d$. The dataset includes distinct node subsets for training, validation, calibration, and testing: $\mathcal{V}^k_{\text{train}}$, $\mathcal{V}^k_{\text{valid}}$, $\mathcal{V}^k_{\text{calib}}$, and $\mathcal{V}^k_{\text{test}}$, respectively.

Assumption 1. *Let S be a global non-conformity score function learned in a federated setting, designed to be permutation invariant with respect to the calibration and test nodes within each client. For any permutation π_k of client k's calibration and test nodes, the non-conformity scores satisfy:*

$$\{S(x_v, y_v) : v \in \mathcal{V}^k_{calib} \cup \mathcal{V}^k_{test}\} = \{S(x_{\pi_k(v)}, y_{\pi_k(v)}) : v \in \mathcal{V}^k_{calib} \cup \mathcal{V}^k_{test}\}.$$

Non-conformity scores obtained through GNN training satisfy the above assumption because chosen GNN models are inherently permutation invariant with respect to node ordering. Each local GNN model accesses all node features during training and optimizes the objective function based solely on the training and validation nodes, which remain unchanged under permutation of the calibration and test nodes. Under Assumption 1, we establish the following lemma.

Lemma 1. *Within the transductive learning setting, assuming permutation invariance in graph learning over the unordered graph $\mathcal{G}^k = (\mathcal{V}^k, \mathcal{E}^k)$, the set of non-conformity scores $\{s_v\}_{v \in \mathcal{V}^k_{calib} \cup \mathcal{V}^k_{test}}$ is invariant under permutations of the calibration and test nodes.*

The proof of Lemma 1 is provided in Appendix. Lemma 1 establishes the intra-client exchangeability of calibration and test samples for transductive node classification. Using Lemma 1, we extend the concept of partial exchangeability to federated graph learning.

Assume that the subgraph at client k, $\mathcal{G}^k$, is sampled from a distribution P_k. During inference, a random test node v_{test}, with features and label $(x_{v_{\text{test}}}, y_{v_{\text{test}}})$, is assumed to be sampled from a global distribution Q_{test}, which is a mixture of the client subgraph distributions according to a probability vector p:

$$Q_{\text{test}} = \sum_{k=1}^{K} p_k P_k,$$

which essentially states that v_{test} belongs to client k with probability p_k.

Definition 1 (Partial Exchangeability)**.** *Partial exchangeability in the context of federated learning assumes that the non-conformity scores between a test node and the calibration nodes within the same client are exchangeable, but this exchangeability does not necessarily extend to nodes from different clients.*

Assumption 2. *Consider a calibration set $\{v_i\}_{i=1}^{n_k}$ in client k and a test node v_{test} in the same client. Under the framework of partial exchangeability (Definition 1), the non-conformity scores $s_{v_{test}}$ and $\{s_{v_i}\}_{i=1}^{n_k}$ are assumed to be exchangeable with probability p_k, consistent with Assumption 1. Therefore, v_{test} is partially exchangeable with all calibration nodes within client k.*

This assumption is justified by the properties of our non-conformity score function S, which, as established under Assumption 1, is designed to be permutation invariant within each client's data. This property supports the hypothesis that within a client, the test node and calibration nodes can be considered exchangeable in terms of their non-conformity scores. The limitation to within-client exchangeability is

due to potential differences in data distribution across different clients, which Assumption 1 does not necessarily overcome. This limitation modifies the upper bound of the coverage guarantee, as elucidated in Theorem 1. Details of this assumption can be found in Appendix A.2.

Theorem 1. *Suppose the graph is partitioned across K clients (i.e., K denotes the number of clients in the federated setting), with each client $k \in [K]$ having n_k calibration nodes. Let $N = \sum_{k=1}^{K} n_k$ and assume $p_k = (n_k + 1)/(N + K)$. If the non-conformity scores are arranged in non-decreasing order as $\{S_{(1)}, S_{(2)}, \ldots, S_{(N+K)}\}$, then the α-quantile, $\hat{q}_\alpha$, is the $\lceil (1-\alpha)(N+K) \rceil$-th smallest value in this set. Consequently, the prediction set*

$$C_\alpha(v_{test}) = \{y \in \mathcal{Y} \mid S(x_{v_{test}}, y) \leq \hat{q}_\alpha\}$$

is a valid conformal predictor where:

$$1 - \alpha \leq P\left(y_{test} \in C_\alpha(x_{v_{test}})\right) \leq 1 - \alpha + \frac{K}{N+K}.$$

This theorem ensures that our method achieves at least $(1 - \alpha)$ marginal coverage. The proof is provided in Appendix A.3.

5.2 GENERATING REPRESENTATIVE NODE FEATURES WITH VARIATIONAL AUTOENCODERS

To mitigate the issue of missing neighbor information in federated graph learning, we introduce a novel approach that utilizes VAEs to generate representative node features within each client. These generated features are shared with the central server and then broadcast across clients to complete the local subgraphs, thereby addressing the problem of missing links.

Each client k trains a `VAE` on its local node features $\{x_v\}_{v \in \mathcal{V}^k_{\text{train}}} \subset \mathbb{R}^d$, aiming to capture the underlying distribution P_k of its data. The `VAE` consists of an encoder $q_{\phi_k}(z|x)$ and a decoder $p_{\theta_k}(x|z)$, where $z \in \mathbb{R}^{d'}$ is the latent representation, with $d' < d$. The `VAE` is trained by maximizing the ELBO given in Section 3.3.

After training, each client generates reconstructed node features by passing its original node features through the encoder and decoder:

$$z_v = q_{\phi_k}(x_v), \quad \tilde{x}_v = p_{\theta_k}(z_v), \quad \forall v \in \mathcal{V}^k_{\text{train}}.$$

Next, K-Means clustering [Kodinariya et al., 2013] is applied to the reconstructed node features $\{\tilde{x}_v\}$ to identify M_k cluster centers $\{c^k_m\}_{m=1}^{M_k} \subset \mathbb{R}^d$:

$$c^k_m = \frac{1}{|C^k_m|} \sum_{\tilde{x}_v \in C^k_m} \tilde{x}_v,$$

where C^k_m is the set of reconstructed node features assigned to cluster m in client k. The number of clusters M_k is determined experimentally through hyperparameter tuning.

The cluster centers $\{c^k_m\}$ are then used as prototype features and shared with the central server. The server aggregates the prototype features from all clients and broadcasts them back to each client. This process allows clients to augment their local subgraphs with representative node features from other clients, effectively approximating the missing neighbor information.

5.3 LINK PREDICTION WITH VGAE FOR MISSING NEIGHBOR COMPLETION

After the generated node features are collected by the central server and broadcast to the clients, we need to predict possible edge formations between the generated nodes and the client subgraphs. To this end, we employ a Variational Graph Autoencoder, effective in graph reconstruction tasks, suitable for our graph completion problem. The `VGAE` model is trained to maximize the ELBO loss.

To ensure that our link prediction model generalizes well across all client subgraphs, we train the `VGAE` in a federated setting using the `FedAvg` [Sun et al., 2022] algorithm. Different client subgraphs may have varying connectivity patterns; thus, the model needs to generalize to diverse subgraphs.

After training, the `VGAE` model is used for link prediction between generated nodes $\hat{X}$ and local subgraph nodes X^k. For each client k, the link prediction process is as follows:

1. **Compute edge probabilities** between generated nodes and local nodes: $\hat{P}^k = \texttt{VGAE}(\hat{X}, X^k)$.
2. **Select the top $p\%$ of edge probabilities** to form new edges: $\mathcal{E}^k := \mathcal{E}^k \cup \left\{(u, v) \mid (u, v) \in \text{Top}_p(\hat{P}^k)\right\}$.
3. **Update the node set and features**: $\mathcal{V}^k := \mathcal{V}^k \cup \hat{\mathcal{V}}, \quad X^k := X^k \cup \hat{X}$.

Here, $\text{Top}_p(\hat{P}^k)$ denotes the set of edges corresponding to the highest $p\%$ of predicted edge probabilities in $\hat{P}^k$. By integrating these new edges and nodes into their local subgraphs, clients enhance their models with previously missing neighbor information. This process is summarized in the Algorithm provided in Appendix B.

Our complete pipeline, which combines generative reconstruction with federated conformal prediction, can be summarized in the following steps:

1. **Local Prototype Generation:** Each client trains a local VAE on its node features to extract and cluster representative feature prototypes.
2. **Server Aggregation:** Cluster centers (prototypes) are

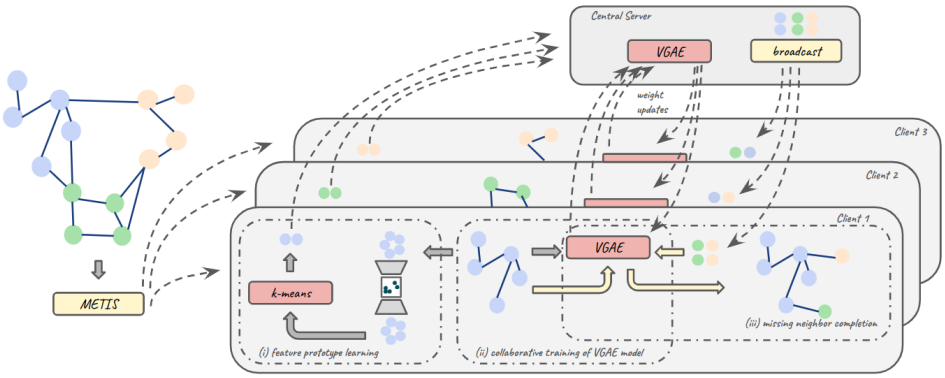

Figure 3: **Missing neighbor generation framework.** *(i) Feature Prototype Learning:* We train VAEs on local subgraph features and apply K-means clustering to obtain prototype node features. The cluster centers serve as feature prototypes, which are sent to the central server for later broadcasting. *(ii) Collaborative Training of VGAE:* We train VGAE models in a federated manner to learn generalizable connectivity patterns across client subgraphs. *(iii) Missing Neighbor Completion:* The central server broadcasts the learned feature prototypes, which are then used to complete missing neighbors via the trained VGAE model.

sent to the server, which aggregates them and broadcasts the global set of prototypes back to all clients.

3. **Collaborative Link Prediction:** A VGAE is trained via FedAvg to learn generalizable connectivity patterns.
4. **Local Subgraph Completion:** Each client uses the global prototypes and the trained VGAE to predict and add missing edges to its local subgraph.
5. **Federated GCN Training:** A GCN model is trained for node classification via FedAvg on the newly completed subgraphs.
6. **Federated Conformal Prediction:** Clients use the global GCN and a held-out calibration set to compute non-conformity scores and generate prediction sets with a distributed quantile estimation.

6 EXPERIMENTS

We conduct experiments on four real-world datasets to demonstrate the effectiveness of our proposed federated conformal prediction method on graph data with varying numbers of clients.

6.1 EXPERIMENTAL SETUP

We evaluate our method on four widely used graph datasets: Cora, CiteSeer, PubMed [Yang et al., 2016], and Amazon Computers [Shchur et al., 2018]. To simulate a federated learning environment, we partition each graph into clusters of $K = 3$, 5, 10, and 20 using the METIS graph partitioning algorithm [Karypis and Kumar, 1997], which ensures clusters are of similar sizes and minimizes edge cuts between partitions. This partitioning introduces missing links between subgraphs, reflecting real-world scenarios where data is distributed across different clients with incomplete neighbor information.

We implement two-layer local permutation-invariant GCN models with mean pooling and employ the FedAvg algorithm [McMahan et al., 2017] to train the global GNN model. The batch size and learning rate for training local GNNs are set to 32 and 0.01, respectively, using the Adam optimizer. For the VAE and VGAE, we use the official implementations from the PyTorch Geometric package [Fey and Lenssen, 2019], with hidden dimensions of size 64 and 16, respectively. The hyperparameters for percentages and number of clusters were determined through a grid search over $p \in \{0.5, 1, 2, 4, 6, 8, 10, 12\}$ and $M \in \{2, 5, 10, 20\}$. The VAE decoder mirrors the encoder dimensions, while the VGAE utilizes an inner product decoder.

Each local subgraph is divided into training, calibration, and test sets in a 20%/40%/40% ratio. 20% of the training set is used for validation. All experiments are conducted on NVIDIA RTX A5000-24GB GPUs.

As recommended by Lu et al. [2023], we apply temperature scaling to the conformal procedure. We average the locally learned temperatures across clients before initiating the federated conformal procedure.

To estimate the set-valued function C_α within our framework, we compute the quantile of conformal scores $\{s_i^k\}_{i=1}^{n_k}$ for each client $k \in [K]$, where each score $s_i^k = S(x_i^k, y_i^k)$ is distributed across K clients. We employ distributed quantile estimation techniques, which have proven effective in traditional federated conformal prediction settings [Lu et al., 2023]. Specifically, we adopt quantile averaging [Luo et al., 2016] and T-Digest [Dunning, 2021], a quantile sketching algorithm designed for efficient online quantile estimation

Table 3: Conformal prediction (CP) set size comparison on four datasets with partition numbers $K = 3, 5, 10$, and 20 using the APS non-conformity score. Set sizes are presented for confidence levels $1 - \alpha = 0.95, 0.90$, and 0.80. The methods `Loc`, `Fed`, and `Gen` correspond to models trained locally on each client, with standard federated averaging, and with our generative neighbor completion framework, respectively. The corresponding standard deviations are given, averaged over 5 runs.

	Cora				**PubMed**			
Model	$K=3$	$K=5$	$K=10$	$K=20$	$K=3$	$K=5$	$K=10$	$K=20$
`Loc (0.95)`	4.97 ±0.02	5.27 ±0.02	5.53 ±0.02	5.90 ±0.02	1.94 ±0.03	1.97 ±0.03	1.98 ±0.04	2.04 ±0.01
`Fed (0.95)`	4.31 ±0.02	**4.94** ±0.02	5.02 ±0.02	5.79 ±0.02	1.80 ±0.02	1.78 ±0.02	1.83 ±0.01	1.77 ±0.02
`Gen (0.95)`	**4.25** ±0.02	5.09 ±0.02	**4.86** ±0.02	**5.40** ±0.02	**1.72** ±0.02	**1.78** ±0.01	**1.80** ±0.02	**1.69** ±0.02
`Loc (0.90)`	4.12 ±0.01	4.54 ±0.01	4.83 ±0.03	4.99 ±0.03	1.79 ±0.00	1.86 ±0.03	1.90 ±0.03	1.95 ±0.03
`Fed (0.90)`	3.34 ±0.03	4.14 ±0.03	4.32 ±0.02	4.13 ±0.01	1.61 ±0.03	**1.60** ±0.01	1.68 ±0.01	1.53 ±0.02
`Gen (0.90)`	**3.34** ±0.02	**4.10** ±0.02	**3.98** ±0.01	**3.90** ±0.04	**1.55** ±0.01	**1.60** ±0.01	**1.62** ±0.02	**1.49** ±0.01
`Loc (0.80)`	3.17 ±0.01	3.77 ±0.01	3.87 ±0.02	4.14 ±0.02	1.72 ±0.01	1.78 ±0.01	1.80 ±0.01	1.69 ±0.01
`Fed (0.80)`	**2.45** ±0.01	**2.95** ±0.01	2.93 ±0.03	3.17 ±0.03	1.55 ±0.01	1.60 ±0.02	1.62 ±0.00	1.49 ±0.02
`Gen (0.80)`	2.51 ±0.03	2.98 ±0.05	**2.92** ±0.02	**2.88** ±0.03	**1.41** ±0.04	**1.42** ±0.01	**1.45** ±0.03	**1.37** ±0.03
	CiteSeer				**Computers**			
Model	$K=3$	$K=5$	$K=10$	$K=20$	$K=3$	$K=5$	$K=10$	$K=20$
`Loc (0.95)`	4.80 ±0.02	4.95 ±0.02	4.99 ±0.02	4.99 ±0.03	6.18 ±0.03	6.31 ±0.04	6.71 ±0.02	6.45 ±0.02
`Fed (0.95)`	3.89 ±0.04	4.12 ±0.01	**4.19** ±0.04	4.42 ±0.01	5.58 ±0.03	5.96 ±0.02	4.76 ±0.03	6.10 ±0.02
`Gen (0.95)`	**3.72** ±0.02	**4.11** ±0.01	4.55 ±0.02	**4.27** ±0.01	**5.08** ±0.02	**5.86** ±0.03	**4.21** ±0.02	**5.30** ±0.01
`Loc (0.90)`	4.14 ±0.01	4.62 ±0.01	4.73 ±0.03	4.87 ±0.03	5.27 ±0.04	5.26 ±0.04	5.66 ±0.02	5.73 ±0.03
`Fed (0.90)`	3.16 ±0.01	3.09 ±0.02	3.14 ±0.03	3.82 ±0.02	4.78 ±0.04	5.37 ±0.02	4.09 ±0.02	5.46 ±0.01
`Gen (0.90)`	**2.95** ±0.02	**2.96** ±0.01	**3.08** ±0.02	**3.65** ±0.01	**4.59** ±0.03	**5.09** ±0.02	**3.57** ±0.01	**4.69** ±0.01
`Loc (0.80)`	3.35 ±0.01	3.59 ±0.02	3.84 ±0.03	3.98 ±0.02	4.24 ±0.04	4.28 ±0.01	4.55 ±0.03	**3.96** ±0.01
`Fed (0.80)`	2.45 ±0.03	2.22 ±0.02	2.17 ±0.02	2.85 ±0.01	**3.60** ±0.01	4.28 ±0.03	3.39 ±0.01	4.66 ±0.03
`Gen (0.80)`	**2.19** ±0.02	**2.11** ±0.02	**2.14** ±0.03	**2.66** ±0.02	3.62 ±0.03	**3.97** ±0.02	**2.96** ±0.02	3.97 ±0.01

in distributed settings.

We present our main results using the APS non-conformity score and quantile averaging for distributed quantile estimation. APS was chosen for its robust performance, and a detailed comparison with other non-conformity scores (RAPS and LAC) is available in Appendix E. Similarly, our choice of quantile averaging is supported by a comparative study with the T-Digest method in Appendix F. We report the average conformal prediction set sizes across clients after local training of GNN models on each client (denoted as `Loc` in Table 3). This allows us to evaluate the empirical impact of the federated conformal procedure on graph data. The `Fed` entry in the table presents results from the experimental validation of our federated conformal prediction method. Results from our generative framework are indicated as `Gen`.

6.2 MAIN RESULTS AND EFFICIENCY ANALYSIS

Our experimental results, presented in Table 3, consistently demonstrate that federated training (`Fed`) achieves smaller CP set sizes compared to local training (`Loc`). This improvement is particularly pronounced as the number of clients increases. For example, on the PubMed dataset with $K = 20$ clients at a confidence level of $1 - \alpha = 0.95$, the federated approach yields an average CP set size of 1.77 ± 0.02, compared to 2.04 ± 0.01 for local training, indicating a significant reduction in uncertainty.

Examining the trends across rows, we observe the adverse effects of missing links on CP set sizes: as the number of clients increases (left to right along each row), the fragmentation of local graphs exacerbates uncertainty, leading to larger CP sets. This underscores the necessity of mitigating missing edge information in federated settings.

Averaging across datasets and client configurations, transitioning from `Loc` to `Fed` yields a notable 15.4% improvement in CP set sizes. This aligns with our theoretical justification, which enables the federated application of conformal prediction while leveraging information across clients to counteract the impact of missing local edges.

Our missing neighbor generator (`Gen`) further refines the CP set sizes across varying configurations and datasets. This approach effectively reconstructs missing node connections, allowing the conformal predictor to approximate its optimal state, where all edges are present. For instance, in the Computers dataset with $K = 20$ and a confidence level of $1-\alpha = 0.95$, `Gen` achieves a CP set size of 5.30 ± 0.01, improving upon both federated (6.10 ± 0.02) and local training (6.45 ± 0.02).

Overall, our neighbor completion strategy in `Gen` provides an additional 4.9% improvement over `Fed`, cumulatively leading to CP sets that are 19.5% smaller than those obtained with `Loc`. This demonstrates that, despite the chal-

lenges posed by missing links in federated settings, our approach successfully reconstructs latent structures, enhancing predictive efficiency in federated graph learning.

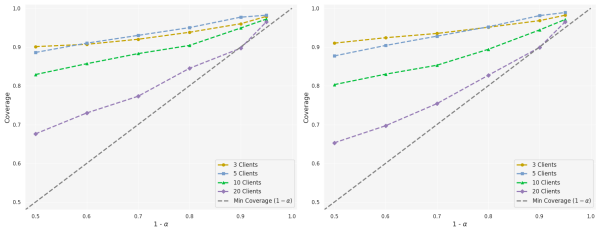

Figure 4: **Coverage Rates** for `Fed` (left) and `Gen` (right) models across varying client numbers K on the Cora dataset. The diagonal line represents the desired coverage rate.

6.3 COVERAGE RATES

As established in our theoretical analysis (Section 5), applying conformal prediction in federated graph learning preserves the lower bound for CP coverage. We empirically validate this on the Cora dataset, as shown in Figure 4, which illustrates the coverage rates across different numbers of clients. The results confirm that our method consistently meets the desired coverage threshold. Notably, for lower $1 - \alpha$ values, the model easily satisfies the required coverage, and its performance remains robust even at more challenging miscoverage rates. This highlights the effectiveness of CP as a reliable uncertainty quantification method in high-stakes applications. Furthermore, incorporating our generative model maintains the desired coverage while significantly reducing CP set sizes, as demonstrated on the right side of Figure 4.

6.4 PARAMETER SENSITIVITY

As shown in Figure 5, varying number of cluster centers consistently improves the set size returned in the `Fed` setting. However, the relationship between the number of clusters and performance does not follow a direct correlation. We hypothesize that due to heterogeneity in client distributions, using a fixed M value across all clients may not always yield the most representative feature prototypes. An adaptive approach to selecting the number of cluster centers per client could potentially enhance performance, which we leave for future work. For the p parameter, we observed that adding a small number of edges to the client subgraphs generally improves performance. However, increasing the number of edges continues to provide benefits in many cases, suggesting that the added edges may act as a form of oversmoothing, making the model less sensitive to noise in the graphs.

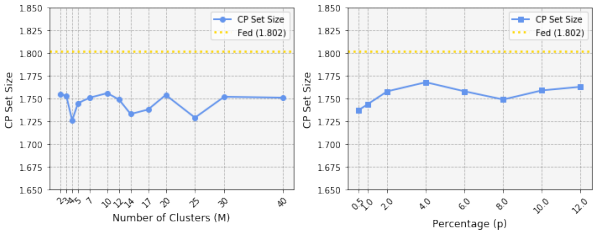

Figure 5: Effect of the number of clusters (M) and percentage (p) on CP set size under the three-client setting on the PubMed dataset. All other parameters are kept constant. The left plot shows CP set size as a function of M, while the right plot shows CP set size as a function of p. The dotted horizontal line represents the baseline CP set size for the federated setting (Fed = 1.802).

7 CONCLUSION

We introduced the first framework for federated conformal prediction for node classification under transductive settings on graphs, addressing two critical challenges: the non-exchangeability of data due to client heterogeneity, and inefficiencies in prediction sets caused by missing inter-client links. We established the theoretical foundations for this problem by extending the principle of partial exchangeability, confirming that reliable marginal coverage guarantees can be achieved. Additionally, we introduced a novel generative model for neighbor reconstruction to address the inefficiencies from missing links. Extensive experiments on real-world graphs demonstrate the practical effectiveness of our method, which maintains valid coverage guarantees while reducing prediction set sizes by up to 19.5% compared to local baselines. Together, these contributions establish a new paradigm for reliable, uncertainty-aware federated graph learning in applications ranging from healthcare networks to distributed social systems.

Acknowledgements

This work is supported by the DEVCOM Army Research Lab (ARL) under grants W911NF2220159 and W911NF2320186.

References

Anastasios Angelopoulos, Stephen Bates, Jitendra Malik, and Michael I Jordan. Uncertainty sets for image classifiers using conformal prediction. *arXiv preprint arXiv:2009.14193*, 2020.

Jinheon Baek, Wonyong Jeong, Jiongdao Jin, Jaehong Yoon, and Sung Ju Hwang. Personalized subgraph federated learning. In *International Conference on Machine Learning*, pages 1396–1415. PMLR, 2023.

Jase Clarkson. Distribution free prediction sets for node classification. In *International Conference on Machine Learning*, pages 6268–6278. PMLR, 2023.

Bruno De Finetti. On the condition of partial exchangeability. *Studies in inductive logic and probability*, 2:193–205, 1980.

Ted Dunning. The t-digest: Efficient estimates of distributions. *Software Impacts*, 7:100049, 2021.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric, 2019.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018. URL `https://arxiv.org/abs/1706.02216`.

Kexin Huang, Ying Jin, Emmanuel Candes, and Jure Leskovec. Uncertainty quantification over graph with conformalized graph neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

George Karypis and Vipin Kumar. Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. 1997.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017. URL `https://arxiv.org/abs/1609.02907`.

Trupti M Kodinariya, Prashant R Makwana, et al. Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95, 2013.

Bhawesh Kumar, Charlie Lu, Gauri Gupta, Anil Palepu, David Bellamy, Ramesh Raskar, and Andrew Beam. Conformal prediction with large language models for multi-choice question answering. *arXiv preprint arXiv:2305.18404*, 2023.

Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

Rui Liu, Pengwei Xing, Zichao Deng, Anran Li, Cuntai Guan, and Han Yu. Federated graph neural networks: Overview, techniques, and challenges. *IEEE transactions on neural networks and learning systems*, 2024.

Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. Geniepath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4424–4431, 2019.

Charles Lu, Yaodong Yu, Sai Praneeth Karimireddy, Michael Jordan, and Ramesh Raskar. Federated conformal predictors for distributed uncertainty quantification. In *International Conference on Machine Learning*, pages 22942–22964. PMLR, 2023.

Ge Luo, Lu Wang, Ke Yi, and Graham Cormode. Quantiles over data streams: experimental comparisons, new analyses, and further improvements. *The VLDB Journal*, 25: 449–472, 2016.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

Yaniv Romano, Matteo Sesia, and Emmanuel Candes. Classification with valid and adaptive coverage. *Advances in Neural Information Processing Systems*, 33:3581–3591, 2020.

Mauricio Sadinle, Jing Lei, and Larry Wasserman. Least ambiguous set-valued classifiers with bounded error levels. *Journal of the American Statistical Association*, 114 (525):223–234, 2019.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Tao Sun, Dongsheng Li, and Bao Wang. Decentralized federated averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4289–4301, 2022.

Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. Fedproto: Federated prototype learning across heterogeneous clients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8432–8440, 2022.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*, volume 29. Springer, 2005.

Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5):1–37, 2022.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

Soroush H Zargarbashi, Simone Antonelli, and Aleksandar Bojchevski. Conformal prediction sets for graph neural networks. In *International Conference on Machine Learning*, pages 12292–12318. PMLR, 2023.

Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. Subgraph federated learning with missing neighbor generation. *Advances in Neural Information Processing Systems*, 34:6671–6682, 2021a.

Xiao-Meng Zhang, Li Liang, Lin Liu, and Ming-Jing Tang. Graph neural networks and their current applications in bioinformatics. *Frontiers in genetics*, 12:690049, 2021b.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

Supplementary Material

Ömer Faruk Akgül[1] **Rajgopal Kannan**[2] **Viktor Prasanna**[1]

[1]University of Southern California, Los Angeles
[2]DEVCOM ARL Army Research Office

A PARTIAL EXCHANGEABILITY PROOFS

A.1 PROOF OF LEMMA 1

Consider the unordered graph $\mathcal{G}^k = (\mathcal{V}^k, \mathcal{E}^k)$ within the permutation-invariant graph learning environment as outlined in the conditions of Lemma 1. Assuming that the graph structure, attribute information, and node label information are fixed, we define the nonconformity scores at nodes in $\mathcal{V}^k_{\text{calib}} \cup \mathcal{V}^k_{\text{test}}$ as

$$\{s_v\} = S\left(\mathcal{V}^k, \mathcal{E}^k, \{(x_v, y_v)\}_{v \in \mathcal{V}^k_{\text{train}} \cup \mathcal{V}^k_{\text{valid}}}, \{x_v\}_{v \in \mathcal{V}^k_{\text{calib}} \cup \mathcal{V}^k_{\text{test}}}\right),$$

where S denotes the scoring function used to compute the nonconformity scores.

Due to the permutation invariance of the model (Assumption 1), for any permutation π of the nodes in $\mathcal{V}^k_{\text{calib}} \cup \mathcal{V}^k_{\text{test}}$, the nonconformity scores remain unchanged. Specifically, we have

$$\{s_v\} = S\left(\pi\left(\mathcal{V}^k\right), \pi\left(\mathcal{E}^k\right), \{(x_v, y_v)\}_{v \in \mathcal{V}^k_{\text{train}} \cup \mathcal{V}^k_{\text{valid}}}, \{x_{\pi(v)}\}_{v \in \mathcal{V}^k_{\text{calib}} \cup \mathcal{V}^k_{\text{test}}}\right).$$

Here, $\pi\left(\mathcal{V}^k\right)$ and $\pi\left(\mathcal{E}^k\right)$ denote the vertex set and edge set permuted according to π.

This invariance implies that, regardless of the permutation of nodes in $\mathcal{V}^k_{\text{calib}} \cup \mathcal{V}^k_{\text{test}}$, the computed nonconformity scores $\{s_v\}$ remain the same. Therefore, the unordered set of scores $\{s_v\}_{v \in \mathcal{V}^k_{\text{calib}} \cup \mathcal{V}^k_{\text{test}}}$ is invariant under permutations of the nodes, confirming the lemma's assertion about the stability and invariance of the score set in this setting.

A.2 REMARK ON ASSUMPTION 2

Under Assumption 2, the nonconformity scores $\{s_{v_i}\}_{v_i \in \mathcal{V}^k_{\text{calib}}}$ for client k are identically distributed and exchangeable. Extending this set to include the score $s_{v_{\text{test}}} = S(x_{v_{\text{test}}}, y_{v_{\text{test}}})$, where $(x_{v_{\text{test}}}, y_{v_{\text{test}}}) \sim P_k$ (the distribution for client k), the augmented set $\{s_{v_i}\}_{v_i \in \mathcal{V}^k_{\text{calib}}} \cup \{s_{v_{\text{test}}}\}$ remains identically distributed and exchangeable.

This demonstrates that $s_{v_{\text{test}}}$ is equivalent in distribution to any s_{v_i} in the calibration set. Therefore, the test score $s_{v_{\text{test}}}$ can be considered as an additional sample from the same distribution, affirming the IID and exchangeability conditions outlined in Assumption 2.

A.3 PROOF OF THEOREM 1

We aim to show that under the given assumptions, the conformal prediction framework achieves the intended coverage guarantees.

Let $N = \sum_{k=1}^{K} n_k$ be the total number of calibration nodes across all clients, where n_k is the number of calibration nodes for client k. Define $p_k = \dfrac{n_k + 1}{N + K}$, so that $\sum_{k=1}^{K} p_k = 1$.

For each client k, let $m_k(q)$ denote the number of nonconformity scores less than or equal to q among the $n_k + 1$ scores (including the test node), that is,

$$m_k(q) = \left|\left\{s_v \mid s_v \leq q,\, v \in \mathcal{V}_{\text{calib}}^k \cup \{v_{\text{test}}\}\right\}\right|.$$

Recall that the conformal quantile $\hat{q}_\alpha$ is defined as the $\lceil (1-\alpha)(N+K) \rceil$-th smallest nonconformity score among all calibration scores and test scores from all clients. Thus,

$$\sum_{k=1}^{K} m_k(\hat{q}_\alpha) = \lceil (1-\alpha)(N+K) \rceil.$$

Define the event $\mathcal{E}$ as the combined ordering of nonconformity scores within each client, that is,

$$\mathcal{E} = \left\{\forall k \in [K],\ \text{the nonconformity scores } \{s_i^k\}_{i=1}^{n_k+1} \text{ are in a fixed order}\right\},$$

where $\{s_i^k\}_{i=1}^{n_k+1}$ are the nonconformity scores for client k, including the test score, sorted in some fixed order.

Conditioned on $\mathcal{E}$, the number of scores less than or equal to $\hat{q}_\alpha$, $m_k(\hat{q}_\alpha)$, is deterministic for each client k.

Under the exchangeability assumption, the probability that the test score $s_{v_{\text{test}}}$ is less than or equal to $\hat{q}_\alpha$ conditioned on $\mathcal{E}$ is

$$P(s_{v_{\text{test}}} \leq \hat{q}_\alpha \mid \mathcal{E}) = \sum_{k=1}^{K} p_k \cdot \frac{m_k(\hat{q}_\alpha)}{n_k + 1}.$$

Therefore, we have

$$P(s_{v_{\text{test}}} \leq \hat{q}_\alpha \mid \mathcal{E}) = \frac{\sum_{k=1}^{K} m_k(\hat{q}_\alpha)}{N+K} = \frac{\lceil (1-\alpha)(N+K) \rceil}{N+K} \geq 1 - \alpha.$$

Similarly, we can derive an upper bound:

$$P(s_{v_{\text{test}}} \leq \hat{q}_\alpha \mid \mathcal{E}) \leq \frac{\sum_{k=1}^{K} (m_k(\hat{q}_\alpha) + 1)}{N+K} = \frac{\lceil (1-\alpha)(N+K) \rceil + K}{N+K} \leq 1 - \alpha + \frac{K}{N+K}.$$

Thus, we have established that the coverage probability satisfies

$$1 - \alpha \leq P(s_{v_{\text{test}}} \leq \hat{q}_\alpha \mid \mathcal{E}) \leq 1 - \alpha + \frac{K}{N+K}.$$

Since $\mathcal{E}$ has probability 1 (it conditions on the ordering which is always possible), the unconditional probability satisfies the same bounds. This completes the proof that the conformal predictor maintains the desired coverage level under the partial exchangeability and permutation invariance assumptions in the graph-structured federated learning setting.

B MODEL DETAILS AND DETAILED ALGORITHM

The subsequent sections detail the algorithms employed in our proposed methodology, encompassing node generation, edge formation, and the application of CP to federated node classification tasks.

Algorithm 1 Federated Graph Learning with Missing Neighbor Generation and Conformal Prediction

Require: K: Number of clients
$\{(\mathcal{V}^k_{\text{train}}, X^k_{\text{train}}, \mathcal{E}^k)\}_{k=1}^K$: Local datasets
M_k: Number of clusters per client
$p\%$: Top percentage for edge selection
R: Number of federated rounds
Learning rates, other hyperparameters
Ensure: Prediction sets $\{C_\alpha(x)\}$ for test nodes across clients

Step 1: Generate prototype node features
for each client $k = 1$ to K **in parallel do**
    Train VAE $q_{\phi_k}(z|x)$, $p_{\theta_k}(x|z)$
    Reconstruct features $\tilde{x}_v = p_{\theta_k}(q_{\phi_k}(x_v))$
    Cluster $\{\tilde{x}_v\}$ into M_k centers $\{c^k_m\}$
    Send $\{c^k_m\}$ to the server
end for

Step 2: Aggregate and broadcast prototypes
Aggregate $\hat{X} = \bigcup_{k=1}^K \{c^k_m\}$
Broadcast $\hat{X}$ to all clients

Step 3: Federated training of VGAE
Initialize global VGAE parameters Θ
for each round $r = 1$ to R **do**
    for each client $k = 1$ to K **in parallel do**
        Receive Θ
        Train local VGAE $q_{\psi_k}(Z|X^k, \mathcal{E}^k)$, $p_{\varphi_k}(\mathcal{E}^k|Z)$
        Send updated Θ_k to server
    end for
    Aggregate $\Theta \leftarrow \frac{1}{K}\sum_{k=1}^K \Theta_k$
end for

Step 4: Link prediction and graph update
for each client $k = 1$ to K **do**
    Compute edge probabilities $\hat{P}^k = \text{VGAE}_\Theta(X^k, \mathcal{E}^k)$
    Select top $p\%$ edges to form new set $\hat{\mathcal{E}}^k$
    Update $\mathcal{E}^k \leftarrow \mathcal{E}^k \cup \hat{\mathcal{E}}^k$
end for

Step 5: Federated GCN training
Initialize global GCN parameters θ
for each round $r = 1$ to R **do**
    for each client $k = 1$ to K **in parallel do**
        Receive θ
        Train local GCN on $(\mathcal{V}^k_{\text{train}}, X^k, \mathcal{E}^k)$
        Send updated θ_k to server
    end for
    Aggregate $\theta \leftarrow \sum_{k=1}^K \frac{n_k}{n}\theta_k$
end for

Step 6: Federated Conformal Prediction
for each client $k = 1$ to K **do**
    Use global GCN to compute predictions $\mu(x)$ and non-conformity scores
    Tune temperature T based on validation data
    Compute local conformal quantile q^k from calibration scores and share with the server
end for
Aggregate quantiles on the server to compute global quantile q
Construct prediction sets $C_\alpha(x)$ for test data using q

B.1 SPARSITY REGULARIZATION FOR NODE FEATURE GENERATION

In addition to the standard reconstruction and KL-divergence losses in the VAE, we incorporate a sparsity regularization term to encourage the generated node features to reflect the sparse nature of real-world graph data. This is crucial for datasets where most node features are inherently sparse, ensuring that the latent representations and reconstructed features remain close to the original sparse structure.

Given the latent representations $z \in \mathbb{R}^{d'}$, the sparsity regularization is applied to the encoder activations to control the average activation levels across the latent dimensions. Let $\hat{\rho} \in \mathbb{R}^{d'}$ denote the mean activation of the latent variables z over all nodes, defined as:

$$\hat{\rho}_i = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} z_{v,i}, \quad \forall i \in [1, d'].$$

We introduce a sparsity target $\rho \in (0, 1)$ that specifies the desired level of activation for each latent variable. The sparsity loss $\mathcal{L}_{\text{sparse}}$ is then defined as the Kullback-Leibler divergence between the desired activation ρ and the average activation $\hat{\rho}$:

$$\mathcal{L}_{\text{sparse}} = \sum_{i=1}^{d'} \left(\rho \log \frac{\rho}{\hat{\rho}_i} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_i} \right).$$

This loss term encourages the activations to stay close to the sparsity target ρ, penalizing deviations from this target. A scaling factor β is used to adjust the contribution of this term, and the overall loss function for training the VAE becomes:

$$\mathcal{L} = \lambda_{\text{rec}} \mathcal{L}_{\text{rec}} + \lambda_{\text{kl}} \mathcal{L}_{\text{kl}} + \beta \mathcal{L}_{\text{sparse}},$$

where $\mathcal{L}_{\text{rec}}$ is the reconstruction loss, $\mathcal{L}_{\text{kl}}$ is the KL-divergence loss, and λ_{rec}, λ_{kl}, and β are weights controlling the relative importance of each term.

Incorporating this sparsity regularization helps ensure that the generated node features remain representative of the original sparse input data, improving the quality and fidelity of the reconstructed features in graph-based learning tasks.

C COMPLEXITY ANALYSIS

In this section, we provide a complexity analysis of the proposed method, focusing on the communication overhead between the clients and the central server, as well as the computational cost related to the exchange and utilization of generated node features.

C.1 PROTOTYPE SHARING COMPLEXITY

After training the VAE, each client k identifies M_k cluster centers, representing the prototype features that will be shared with the central server. The dimensionality of each prototype is d, and the total communication cost of sending the prototype features from client k to the server is:

$$\mathcal{O}(M_k \cdot d).$$

Since there are K clients in total, the overall communication complexity for sending prototypes to the server is:

$$\mathcal{O}(K \cdot M_k \cdot d),$$

where M_k may vary across clients but is typically constant for simplicity.

C.2 SERVER AGGREGATION COMPLEXITY

The central server aggregates the prototype features from all clients, combining them into a global set of features $\hat{X} = \bigcup_{k=1}^{K} \{c_m^k\}$. This aggregation step involves concatenating the received prototypes, which has a complexity of:

$$\mathcal{O}(K \cdot M_k \cdot d).$$

The server then broadcasts the aggregated prototypes back to all clients. The communication complexity of broadcasting the prototypes from the server to all clients is:

$$\mathcal{O}(K \cdot M_k \cdot d),$$

assuming all clients receive the same set of $(K-1) \cdot M_k$ prototypes. Thus, the total communication cost for the prototype-sharing phase (sending prototypes to the server and broadcasting them back) is:

$$\mathcal{O}(2 \cdot K \cdot M_k \cdot d).$$

C.3 FEDERATED TRAINING COMMUNICATION COMPLEXITY

During the federated training of the `VGAE` model, each client k sends its local model updates Θ_k to the central server. The model parameters Θ_k are of size $|\Theta|$, which is the same across all clients. The communication complexity for each client sending its updated model to the server is:

$$\mathcal{O}(|\Theta|).$$

The server aggregates the model updates from all K clients, which involves summing the model parameters. The complexity of this aggregation step is:

$$\mathcal{O}(K \cdot |\Theta|).$$

The server then sends the updated global model back to each client, with a communication complexity of:

$$\mathcal{O}(K \cdot |\Theta|),$$

since each client receives the full set of model parameters. Thus, the total communication complexity for one round of federated training is:

$$\mathcal{O}(2 \cdot K \cdot |\Theta|).$$

C.4 OVERALL COMMUNICATION COMPLEXITY

The overall communication complexity of the proposed method consists of two main components: (1) prototype sharing and (2) federated training. The total communication complexity is the sum of these two components, which can be expressed as:

$$\mathcal{O}(2 \cdot K \cdot M_k \cdot d + 2 \cdot K \cdot |\Theta| \cdot R).$$

This complexity scales linearly with the number of clients K, the number of prototypes M_k, the number of training epochs R, and the size of the model $|\Theta|$. Therefore, the communication overhead remains manageable, even as the number of clients and the model size increase.

D DATASETS STATISTICS

We used the largest connected components of Cora, CiteSeer, PubMed [Yang et al., 2016], and Amazon Computers [Shchur et al., 2018] datasets in the Pytorch Geometric package [Fey and Lenssen, 2019]. Dataset statistics are given in Table 4.

Table 4: Dataset statistics.

Dataset	# Nodes	# Edges	# Features	# Labels
Cora	2485	10138	2485	7
CiteSeer	2120	7358	3703	6
PubMed	19717	88648	500	3
Computers	13752	491722	767	10

E COMPARISON OF NON-CONFORMITY SCORES

Regularized Adaptive Prediction Sets (RAPS) [Angelopoulos et al., 2020] refine APS by introducing regularization to penalize less likely labels. RAPS modifies the score function to include a regularization term, encouraging smaller prediction sets. The score function is defined as

$$s(x, y) = -(\rho(x, y) + u \cdot \pi(x)y + \nu \max(o(x, y) - k, 0))$$

, where ν and k are hyperparameters, and $o(x, y)$ represents the rank of y.

Least Ambiguous Set-Valued Classifiers (LAC) [Sadinle et al., 2019]assess classification uncertainty. The classifier's score, $s(x, y)$, is given by:

$$s(x, y) = 1 - [f(x)]_y$$

where $[f(x)]_y$ represents the score of the true label, thus quantifying the classifier's confidence in its prediction.

Table 5 compares the CP set sizes when using APS, RAPS, and LAC as the non-conformity scores. While our proposed generative model improves efficiency across all three, LAC consistently yields the smallest prediction sets in most scenarios. However, a known trade-off exists between set size and coverage reliability, as noted in prior work. Our findings confirm this trade-off: Figure 6 shows that while LAC produces tighter sets, it increasingly violates the desired $1 - \alpha$ coverage guarantee as the number of clients grows. Because APS and RAPS reliably maintain the target coverage, they are preferable for high-stakes applications.

Table 5: CP set size comparison of non-conformity scores APS, RAPS and LAC on Cora dataset with partition number $K = 3, 5, 10$ and 20. Set sizes are presented for $1 - \alpha = 0.95, 0.90$, and 0.80 confidence levels. The corresponding std. are given with an averaged set size over 10 runs.

	APS	RAPS	LAC	APS	RAPS	LAC
	$K = 3$			$K = 5$		
Fed (0.95)	4.31±0.02	2.57±0.02	**1.79**±0.01	4.94±0.02	2.97±0.01	**2.59**±0.03
Gen (0.95)	4.25±0.02	2.22±0.01	**1.58**±0.02	5.09±0.02	2.82±0.02	**2.53**±0.04
Fed (0.90)	3.34±0.03	1.85±0.01	**1.19**±0.01	4.14±0.03	2.33±0.02	**1.64**±0.02
Gen (0.90)	3.34±0.02	1.69±0.02	**1.12**±0.01	4.10±0.02	2.12±0.03	**1.61**±0.01
Fed (0.80)	2.45±0.01	1.36±0.01	**1.01**±0.01	2.95±0.01	1.63±0.01	**1.04**±0.00
Gen (0.80)	2.51±0.03	1.27±0.02	**1.00**±0.00	2.98±0.05	1.52±0.02	**1.04**±0.00
	$K = 10$			$K = 20$		
Fed (0.95)	5.02±0.02	**3.50**±0.01	3.82±0.02	5.79±0.02	**5.16**±0.02	5.64±0.01
Gen (0.95)	4.86±0.02	**3.39**±0.03	3.39±0.04	5.40±0.02	**4.92**±0.05	5.06±0.03
Fed (0.90)	4.32±0.02	2.61±0.01	**2.06**±0.01	4.13±0.01	3.78±0.01	**3.37**±0.00
Gen (0.90)	3.98±0.01	2.55±0.02	**2.00**±0.01	3.90±0.04	3.55±0.03	**3.05**±0.01
Fed (0.80)	2.93±0.03	1.79±0.01	**1.19**±0.00	3.17±0.03	2.92±0.01	**2.27**±0.01
Gen (0.80)	2.92±0.02	1.73±0.01	**1.14**±0.02	2.88±0.03	2.50±0.01	**1.73**±0.03

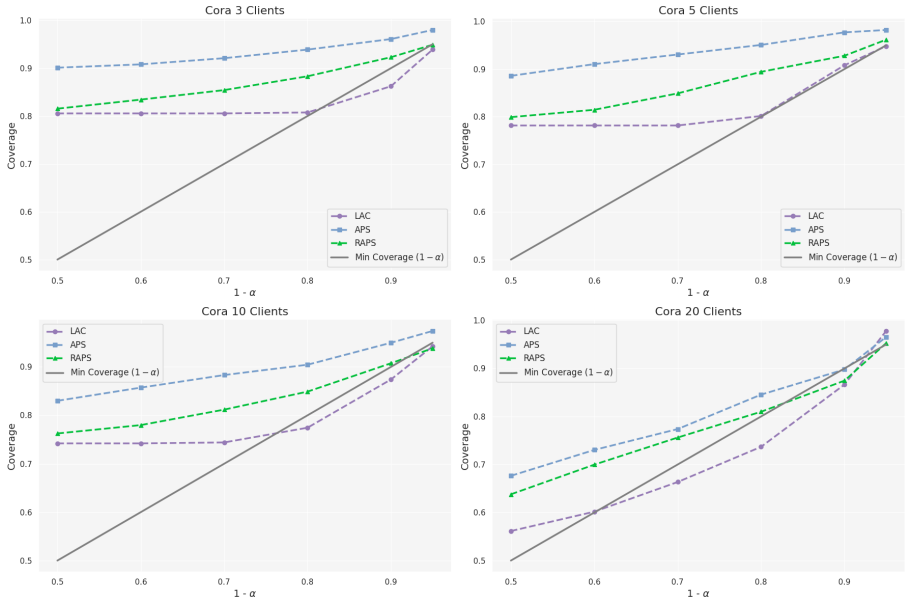

Figure 6: **Coverage rates with different non-conformity scores for `Fed` model across varying K on the Cora dataset.**

F IMPACT QUANTILE AVERAGING METHODS

In this study, we evaluated the performance of two distributed quantile estimation methods, T-Digest [Dunning, 2021] and quantile averaging [Luo et al., 2016], with respect to their impact on conformal prediction set sizes. T-Digest is a probabilistic data structure optimized for the estimation of quantiles in extensive and distributed datasets, facilitating real-time analysis. Its mergeable nature enables effective aggregation of summaries across parallel, distributed systems, ensuring statistical efficiency and scalability. As shown in Figure 7, T-Digest produces larger set sizes across various configurations of confidence levels and number of clients. We found quantile averaging is more effective at reducing model uncertainty.

G DIFFERENTIAL PRIVACY ANALYSIS

In our framework, we apply ϵ-δ differential privacy (DP) [Dwork et al., 2014] specifically to the node prototypes generated by the VAE before they are shared with the server. The subsequent federated training of the downstream GCN and VGAE models is performed without DP guarantees; therefore, we do not claim end-to-end privacy for the entire system. Thanks to the post-processing immunity property of DP, any downstream use of the DP-protected prototypes does not degrade the initial privacy guarantee. This section explores the integration of DP into the node generation process.

Under DP, a randomized mechanism $\mathcal{M}$ satisfies ϵ-δ privacy if for any two neighboring datasets D and D', the following holds:

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\epsilon} \Pr[\mathcal{M}(D') \in S] + \delta,$$

where $\epsilon > 0$ controls the privacy loss and δ accounts for the probability of a privacy breach.

The node generator model is trained using the Opacus library to implement privacy-preserving stochastic gradient descent (DP-SGD), which ensures that each client's data is protected by clipping gradients and adding Gaussian noise. This technique introduces an additional noise term to the training process, making it difficult for an adversary to infer individual node features while still enabling useful feature generation. The impact of DP on model performance is explored by varying the privacy budget ϵ and fixing $\delta = 10^{-5}$.

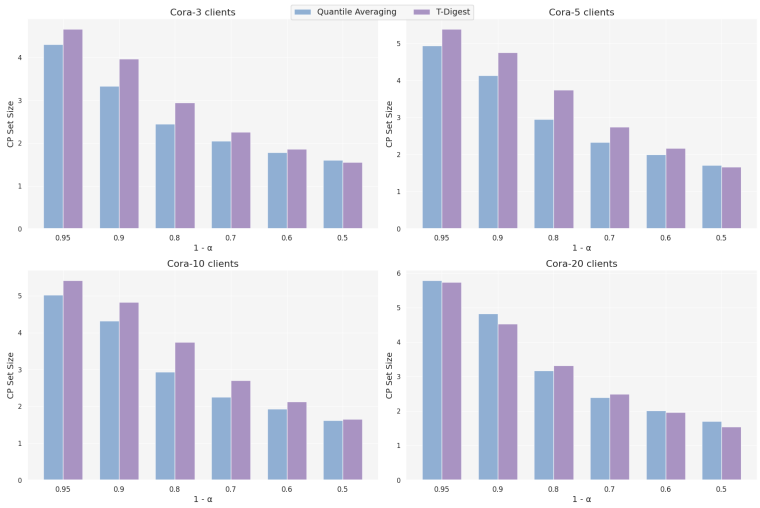

Figure 7: **Comparison of T-Digest and quantile averaging methods by confidence level on Cora dataset.**

G.1 PERFORMANCE WITH VARYING PRIVACY BUDGETS

We evaluate the effectiveness of our node generation method under different privacy budgets by training the `Gen` method with ϵ values ranging from 1 to 25. We analyze the impact of privacy noise on the RAPS non-conformity scores for various $1-\alpha$ values (ranging from 0.5 to 0.95), comparing the results against the non-private `Fed` and `Gen` methods.

Figure 8 presents the observed scores. We note that as the privacy budget decreases (i.e., smaller ϵ values), the performance of the `Gen` method degrades slightly, particularly for larger $1-\alpha$ values. This degradation is expected due to the additional noise introduced by the DP mechanism, which affects the accuracy of the generated node features. However, even with $\epsilon = 1$, the degradation remains relatively small, demonstrating that our approach maintains robust performance under strict privacy constraints.

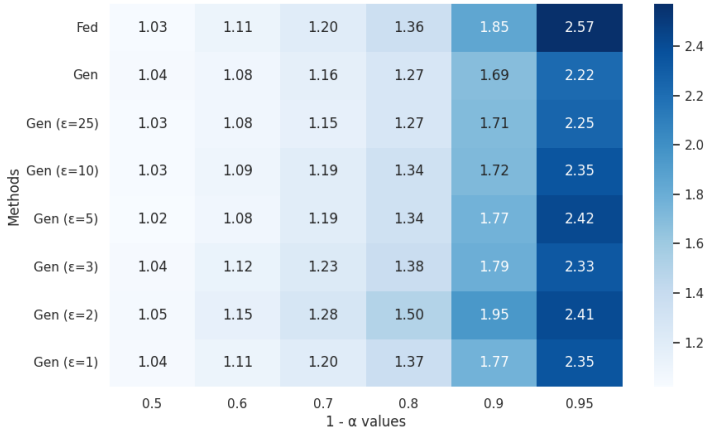

Figure 8: **Heatmap showing RAPS non-conformity scores for `Fed` and `Gen` methods across various ϵ-values and $1-\alpha$ values on 3 client Cora dataset.**

From the results, we observe that at $\epsilon = 10$, the privacy-preserving `Gen` model closely approximates the performance of the non-private `Gen` method across all $1-\alpha$ values. However, with stricter privacy budgets (e.g., $\epsilon = 1$), there is a marginal increase in non-conformity scores, indicating a slight decrease in accuracy due to the added noise. Despite this, the model remains competitive even under the strictest privacy constraints.

Our experiments show that incorporating ϵ-δ differential privacy into the node generation process enables strong privacy guarantees with minimal impact on performance. Even under the strictest privacy settings, the model retains its ability to generate useful node features, as evidenced by the modest increases in RAPS non-conformity scores.