Federated Learning on Non-IID Graphs

Anjana Yodaiken
Department of Computer Science
University of Cambridge
agmy2@cam.ac.uk

Bao Nguyen
Department of Computer Science
University of Cambridge
btn21@cam.ac.uk

Riccardo Conci
Department of Computer Science
University of Cambridge
rc667@cam.ac.uk

Abstract

Federated Learning (FL) has emerged as a privacy-preserving solution for machine learning (ML). Despite the increasing popularity of FL and its application to more common network architectures, there is limited research into federated Graph Neural Networks (GNNs). Specifically, this project investigates the effectiveness of popular FL optimisation algorithms under various conditions of graph heterogeneity. We perform 534 federated simulations across three graph datasets across varying levels of heterogeneity. We show the benefit of FedProx compared to other popular FL optimisation methods in heterogeneous settings induced by the partitioning of node features across clients. Furthermore, we show the shortcomings of Krum in heterogeneous settings created by data poisoning.

The code can be found at: https://github.com/baon6052/federated_learning_gnn.

1 Introduction

Federated Learning (FL) [1] has emerged as a privacy-preserving solution for machine learning (ML). It proposes a paradigm where decentralised data sources, each holding local sensitive data, collaboratively train an ML model without the need to exchange data between them. This naturally gives rise to a class of more intelligent models, as training can be performed on data that was otherwise restricted. FL has been applied in a variety of settings, including healthcare [2–4], recommender systems [5] and secure finance [6, 7].

Additionally, FL can be applied to scenarios that can best be represented by graphs. This includes molecular modelling [8, 9], traffic forecasting [10, 11] and natural language processing [12, 13]. Graph Neural Networks (GNNs) are ML models trained on graphical data. This enables the model to learn the complex relationships between entities represented in the graph.

In this report, we investigate the efficacy of different FL optimisation strategies in mitigating the effects of data heterogeneity in federated graph settings. We do this across two experiments:

1. *Node feature heterogeneity*: we determine the robustness of common FL strategies in various heterogeneous settings. These settings are simulated by partitioning node features across clients and adjusting the amount of shared features.
2. *Data poisoning attacks*: we perform label-flipping as untargeted attacks on clients in an FL setting. We test optimisation strategies that the literature suggests are robust to Byzantine attacks and compare their performance to a simpler algorithm, FedAvg [1].

2 Background

2.1 Federated Learning

A standard FL setting consists of numerous clients that collaboratively train a global model. For a single federated round, each client $k \in K$ independently trains a local model on its own private dataset $d_k \in D$ and sends its local model parameters to the central server. The central server aggregates the clients' parameters. These aggregated parameters are then used to update the global model before distributing the updated global model back to the clients. Intuitively, the aggregation process can be thought of as collecting the learnt information from each client to produce a global understanding across all datasets in D. The objective function can be seen in Equation (1).

$$\min_{w} f(w) = \sum_{k=1}^{|K|} p_k F_k(w) = \mathbb{E}_k[F_k(w)] \tag{1}$$

The term p_k is a weighting coefficient, indicating the importance of the learnt model parameters for client k. This is often defined as the ratio $\frac{n_k}{n}$, where n_k and n are the number of samples assigned to client k and the total number of samples in the training dataset, respectively. F_k is the local loss function of client k, which measures its empirical risk over its local data distribution d_k.

2.2 Graph Neural Networks

A graph $G = (V, E)$ is composed of a set of nodes V and a set of edges $E \subset V \times V$. Two nodes $u, v \in V$ are connected if an edge exists between them, i.e. $(u, v) \in E$. In GNNs, each node in the graph is uniquely defined as a d-dimensional feature vector $h_v \in \mathbb{R}^d$, which expresses the node's attributes. GNNs learn the graph's representations by computing a function, ϕ, known as *message passing*. Starting with an initial node representation $h_v^{(0)}$ for node v, subsequent representations are generated by incorporating the neighbourhood N_v. More formally, this can be defined by Equation (2).

$$\mathbf{h}_v^{(l+1)} = \phi\left(\mathbf{h}_v^{(l)}, \bigoplus_{u \in \mathcal{N}_v} \psi\left(\mathbf{h}_v^{(l)}, \mathbf{h}_u^{(l)}\right)\right) \tag{2}$$

$\bigoplus$ is a permutation-invariant function, such as the summation or mean, to ensure the aggregation of neighbouring node features is invariant to the order of nodes. ψ is a readout function. Both seen in practice, ϕ and ψ are multi-layer perceptrons.

2.3 Federated Graph Neural Networks

Two primary settings exist for training federated GNNs (FedGNNs) on graph data [14–17]: horizontal and vertical. In a horizontal FedGNN setting, graph topologies differ between clients, as shown in Figure 1. There exist two variants:

1. When the server knows if there are missing edges between the clients' graphs and can use this knowledge to easily reconstruct a global graph.
2. When the server does not know if there are missing edges. The first objective is to reconstruct the global graph before federated training. This is achieved by each client sending their encrypted graph data to the server. [18–20]

In a vertical FedGNN setting, there exist two variants:

1. When each client holds a dimension of the graph data, such as node features, relationship data and node labels.
2. When data from the same dimension is shared across clients. i.e. 200 features from the original dataset are shared in batches of 100, 50 and 50 across three clients, as shown in Figure 2.

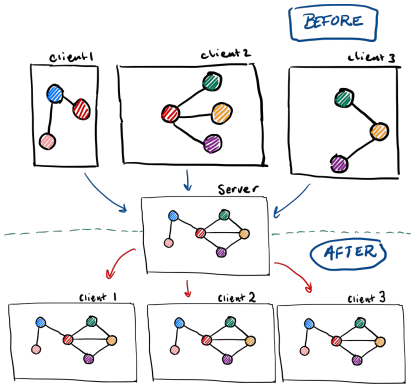


Figure 1: Graph completion before federated training in horizontal FedGNNs

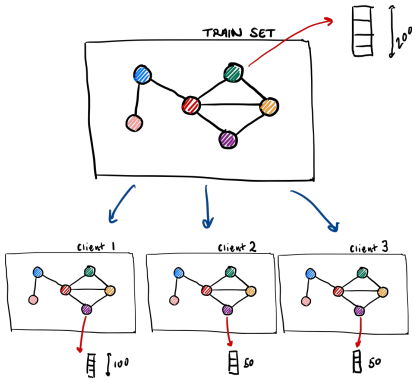


Figure 2: Node feature splitting in vertical FedGNNs

2.3.1 Federated Optimisation Algorithms

Different strategies are used in FL to aggregate local model parameters to produce a global model. FedAvg [1], the simplest yet effective method, averages the client models' parameters. However, its performance declines with non-IID data through weight divergence. To address this, FedProx adds a proximal term to the clients' objective function, penalising significant deviations from the global model. Building on FedProx [21], FedNova [22] introduces a normalising factor that scales client gradients based on their update frequency, addressing slow convergence and objective consistency. Additionally, adaptive optimisers such as FedOpt [23], FedAdam [23], FedAdagrad [23], and FedYogi [23] have shown effectiveness in handling heterogeneity, especially in cross-device settings. In these experiments, we evaluate FedAvg, FedProx, FedOpt, FedAdam, FedAdagrad and FedYogi.

2.4 Poisoning

In FL, "poisoning attacks" involve deliberate interference with the global model's training. These attacks can occur through data tampering or by altering model parameters. FL is particularly vulnerable to such attacks due to the concealment of client data and training processes from the central server. Although these manipulations can happen on the client side, in the communication channel, or on the server, the focus here is on client-side disruptions. [24, 25]

2.4.1 Model poisoning attacks

In model poisoning attacks (MPA), attackers disrupt federated training by "poisoning" local parameters sent from clients to servers. Attackers employ various methods to enhance attack efficacy and reduce detectability. These include gradient manipulation, where attackers directly alter client weights before they are passed to the server. To reduce detectability, the attacker can employ optimisation methods that limit the client weight changes while still manipulating the model towards attacker-specified outcomes. Untargeted attacks focus on degrading the global model's overall accuracy without aiming for specific attacker-defined output. [25, 26]

2.4.2 Data poisoning attacks

In a data poisoning attack (DPA), a client's training data is altered. This affects the training of the client model and, ultimately, results in a MPA. Two broad categories of DPAs include backdoor attacks and untargeted attacks. In backdoor attacks, the attacker injects triggers into training data to subtly degrade subtask performance while maintaining the main task accuracy. Maintaining the main task accuracy makes these types of attacks hard to detect. Untargeted attacks aim to broadly disrupt model performance, either by hindering training convergence or exploiting FL framework structures by masquerading as a client. Label-flipping attacks involve altering data labels, either targeting specific labels or randomly flipping. Poisoning sample attacks insert patterns or noise into training data and as a result, are only appropriate for continuous data. [25, 27, 28]

2.4.3 Defense strategies

Various defence strategies aim to detect or mitigate the effects of poisoning attacks. These include *anomaly detection*, *robust aggregation* and *perturbation mechanisms* [25]. In this project, we will focus on several *robust aggregation* strategies. These include FedMedian [29], FedTrimmedAvg [29] and Krum [30] implemented by the Flower (flwr) Framework [31]. FedMedian uses the median of the updates from all nodes, making it robust to outliers or in this case poisoned client updates. FedTrimmedAvg trims a percentage of the highest and lowest values of the updates from all nodes. The mean is calculated on the remaining data, ideally removing the effects of the contaminated client updates. Krum uses a single most reliable client update or vector for the global model update. This update is identified by finding the vector with the minimum sum of the pairwise distance between it and its k-nearest neighbours. This algorithm assumes that most clients are uncorrupted, thereby grouping the 'reliable' vector with other uncorrupted vectors and effectively ignoring corrupted ones.

3 Methods

To explore the efficacy of various federated optimisation strategies in settings with different levels of client heterogeneity, we perform two experiments, totalling 534 experimental runs architected on the Flower framework [31].

1. The *node feature heterogeneity experiments* in Section 3.1 explore how optimisation strategies best deal with non-IIDness in a horizontal FedGNN setting, i.e. where the features attached to each node in a GNN are split across clients.
2. The *data poisoning attack experiments* in Section 3.2 explore how various optimisation strategies deal with malicious non-IIDness created by untargeted label-flipping attacks.

Datasets: The node feature heterogeneity experiments were run on the Cora [32], CiteSeer [33] and PubMed [34] datasets, whilst only Cora [32] was used in the poisoning experiments. Details on these datasets are found in Table 1.

Models: Graph convolutional networks (GCN) [35] and Graph Attention Networks (GAT) [36] were hyperparameter-tuned across all datasets. Given the increased performance of GAT networks compared to GCN networks, as shown in Table 4, both experiments were run on the hyperparameter-optimised GAT networks.

Compute: These experiments were run on three Apple MacBook Pros and one Apple MacBook Air. Due to the lack of GPU cluster access and the lack of Apple Metal GPU support in the PyTorch Geometric GAT model [37], training was performed on M1 and M2 Pro CPUs.

Table 1: Datasets

Dataset	Nodes	Edges	Avg Degree	Features	Classes	Feature Data Type
Cora	2,708	5,429	3.88	1,433	7	Binary
Citeseer	3,327	4,732	2.84	3,703	6	Binary
PubMed	19,717	44,338	4.5	500	3	Positive real-valued

3.1 Node Feature Heterogeneity Experiments

288 experiments (96 per dataset) were run to assess the quality of federated optimisation algorithms across varying levels of client heterogeneity. The parameters for these experiments are detailed on Table 2.

Node Feature Partition Function. Ideally, as the overlap fraction increases, the total number of features per client stays the same and the fraction of those unique features versus shared changes. This allows experiments to isolate test accuracy differences due to changes in heterogeneity. A 1.0 and 0.0 overlap fraction exhibit complete homogeneity and heterogeneity across clients, respectively. Algorithm 12 achieves this aim. The trade-off being that as the overlap fraction increases, the loss of overall features included in training decreases. For example, in a setting with 100 total features and

Table 2: Summary of Experimental Variables for Node Feature Heterogeneity

Experimental Variable	Values
Number of Clients	2, 4, 8, 10
Overlap Fraction	0, 0.25, 0.5, 1
Optimisation Strategies	FedAvg, FedProx, FedOpt, FedAdam, FedYogi, FedAdagrad
Dataset	Cora, CiteSeer, PubMed

10 clients with a 0.5 overlap fraction, each client should receive 10 features (5 shared and 5 unique). As a result, 45 features would not be included in the training of this model.

Algorithm 1: Node Feature Partitioning

```
num_features_per_client ←num_features ÷ num_clients
num_overlap_features ← floor ( num_features_per_client × overlap_fraction)
num_unique_features ← num_features_per_client - num_overlap_features
partitioned_data ← empty list
for index in range(num_clients) do
    start_index ← num_overlap_features + (index × num_unique_features)
    end_index ← start_index + num_unique_features
    unique ← shuffled_features[ : , start_index:end_index]
    partition ← Concatenate(overlap_data, unique)
    partitioned_data ← Concatenate(partitioned_data, partition)
end
return partitioned_data
```

3.2 Poisoning experiments

The data poisoning attacks were performed using GAT, Cora and 10 clients trained for 10 epochs and 50 rounds. Untargeted label-flipping attacks poisoned a fraction of clients, with a fraction of their node features flipped, as shown in Algorithm 2. Table 3 shows the parameters that were varied in the data poisoning experiments.

Table 3: Settings for data poisoning experiments.

Parameter	Values
Fraction of clients poisoned	0.0, 0.1, 0.3, 0.5, 1.0
Fraction of node features to flip	0.01, 0.1, 0.25, 0.5, 1.0
Optimisation strategies	FedAvg, FedMedian, Krum, FedTrimmedAvg

Algorithm 2: Data Poisoning

```
... ← same as Algorithn 1
for i in range(num_clients) do
    ... ← same as Algorithm 1
    if i in poison_client_indices then
        shuffle_indices_poison ← randomly permute indices of partition
        indices_to_poison ← shuffled_indices_poison[: num_features_per_client ×
          data_poison_frac]
        partition[:,indices_to_poison] ← 1.0 - partition[:,indices_to_poison]
    end
    partitioned_data ← Concatenate(partitioned_data, partition)
end
return partitioned_data
```

4 Results & Discussion

Table 4: Baseline GNN performance across Cora, CiteSeer and PubMed.

Dataset	Network	Hidden Features	Hidden Layers	Learning Rate	Test Accuracy
Cora	GCN	8	2	0.0001	0.57
CiteSeer	GCN	16	0	0.01	0.51
Pubmed	GCN	16	0	0.001	**0.78**
Cora	GAT	4	0	0.0001	**0.83**
CiteSeer	GAT	16	0	0.0001	**0.70**
Pubmed	GAT	8	0	0.001	0.77

4.1 Node feature partitioning experiments

Figure 3 shows the overall results of the node feature partitioning experiments. *The left column* plots are constructed by taking the average test accuracies across all clients for each overlap fraction. From a federated perspective, it shows how optimisation strategies manage increasing heterogeneity, regardless of the number of clients. For example, at an overlap fraction of 0.25, the optimisation strategy curves show an average test accuracy across 2, 4, 8 and 10 clients. *The right column* plots show the average test accuracies across all overlap fractions for each number of clients. From a federated perspective, it shows how optimisation strategies deal with increasing numbers of clients regardless of the heterogeneity across those clients. The *middle column* illustrates which optimisation strategy performs best overall and allows us to compare optimisation strategies across datasets.

Optimisation strategies across increasing heterogeneity

The left column of Figure 3 shows the optimisation strategies as the overlap fraction decreases to 0.0 and clients become more heterogeneous. The test accuracy drops across all datasets. This is marked in Cora, with a drop of approximately 17%. At 0.0 overlap fraction, across all datasets, the performance of the optimisation strategies are within 2% of each other. This suggests that no strategy is significantly better than the others at dealing with this extreme heterogeneity.

Optimisation strategies across increasing clients

The right side of Figure 3a shows a 10% accuracy range for 10 clients across optimisation strategies in Cora, 4% in PubMed (Figure 3c) and 3% in CiteSeer (Figure 3b). These results show that the optimisation strategies diverge with an increasing number of clients instead of increasing heterogeneity.

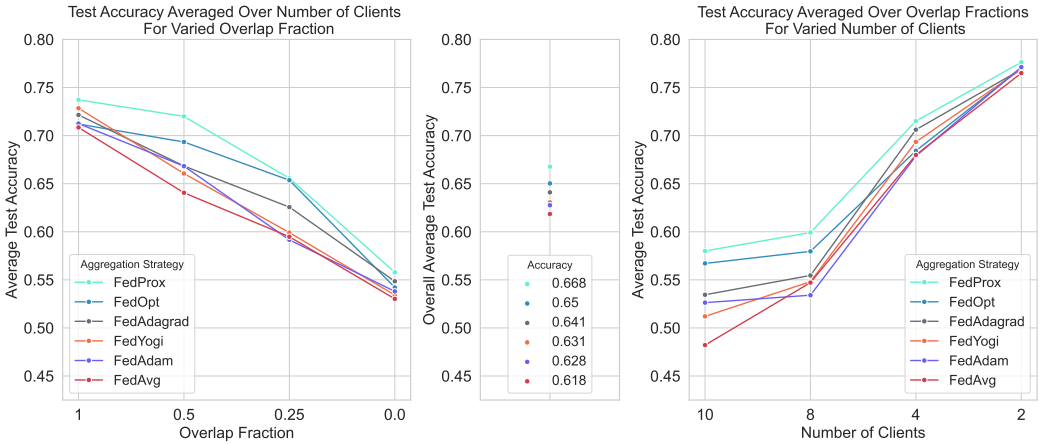
Features in Cora are binary, sparse and few (1,433). CiteSeer has many (3,703) sparse features. PubMed has the fewest (500) but most informative features as they are positive real-valued features in comparison to Cora's and CiteSeer's binary features. Cora's combination of less informative, sparse and few features increases the strain on the optimisation strategies and explains the largest spread across them in Cora.

FedProx exhibits an increased performance on the Cora dataset across 10 clients compared to the other optimisation strategies. The incentive to learn shared features induced by FedProx seems especially important when clients have few and sparse features. Given the similarity of results across PubMed and CiteSeer, differences across optimisation methods are likely not significant.

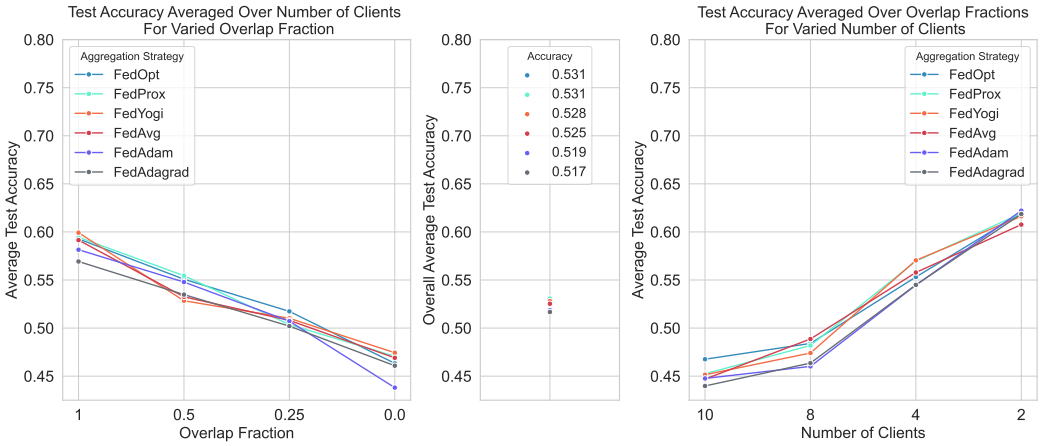
4.2 Data Poisoning Attacks

Figure 4 shows that the model's performance improves as data poisoning decreases, whether in the number of clients or the proportion of data poisoned. This is expected as less poisoned data would typically mean better overall model performance.

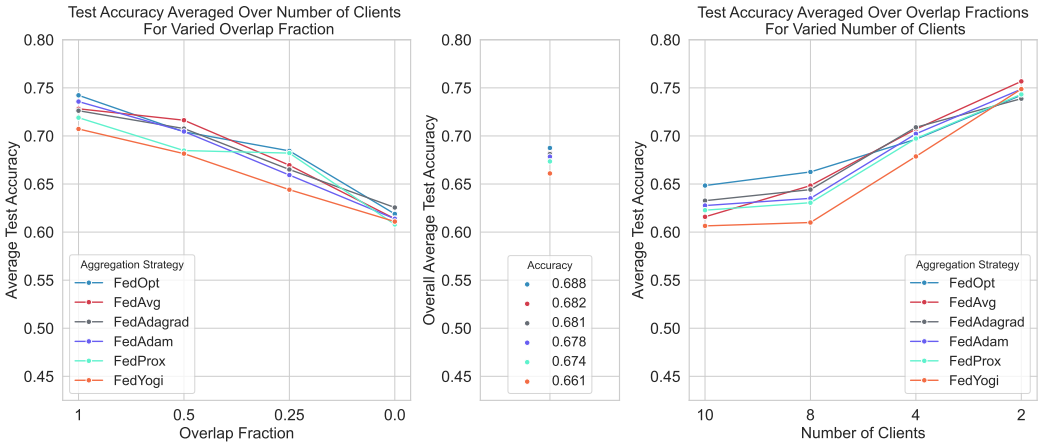
FedMedium, FedTrimmedAvg and FedAvg display comparable performance, with their overall average test accuracies being 61%, 60.9% and 60.3% respectively (within 1%). However, Krum's[30] overall performance is the worst, with an overall average test accuracy of 56.9%, approximately 3% lower than the other optimisation strategies.



(a) Cora Dataset



(b) CiteSeer Dataset



(c) Pubmed Dataset

Figure 3: Left: average test accuracy over number of clients for varied overlap fractions. Middle: average test accuracy across all experiments. Right: average test accuracy for all overlap fractions for varied number of clients.

One would expect that FedMedium and FedTrimmedAvg would outperform FedAvg and that Krum, where the number of honest clients is less than 50%, would perform the best. Although there is a downward trend in Krum's performance, as the client poison fraction increases, Krum consistently performs the worst even for lower client poison fractions.

The similar performance of FedTrimmedAvg, FedMedian, and FedAvg, despite the theoretical robustness of FedTrimmedAvg and FedMedian, suggests that the attacks are not effective enough to skew the average or median to a faulty update, resulting in all three methods performing similarly well. For a similar reason, Krum's performance would degrade if the attacks create malicious client updates that are not distinctly different from the honest client updates. This is because Krum relies heavily on the assumption that the correct model updates are clustered together and the faulty ones exist far away from the honest group.

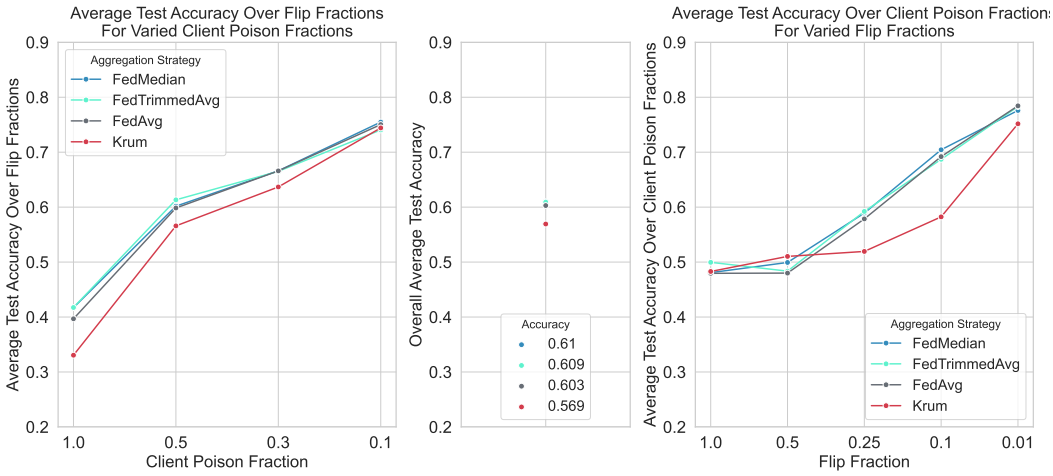


Figure 4: Left: average test accuracy over all flip fractions for varied client poison fractions. Middle: average test accuracy across all experiments. Right: average test accuracy over all client poison fraction for varied flip fractions.

5 Limitations & Future Work

Exhaustively testing optimisation strategies across heterogeneous settings is a significant challenge. The main limitation of our work is the simulated methods by which we induced data heterogeneity and poisoning, limiting the validity of results across settings. Statistical tests would be required to suggest the level of validity of our experiments.

Heterogeneity experiments: future work would include unequal partitioning, data heterogeneity changing over time, splitting by node dimensions, and across horizontal FedGNN strategies. Further works would also expand into model heterogeneity. Furthermore, decentralised FL would be an interesting and novel setting to examine data and model heterogeneity.

Poison experiments: future work would include poisoning that increases over time, using more realistic real-valued feature datasets, and applying different poisoning strategies such as poisoning sample attacks.

6 Conclusion

In this simulated data heterogeneity vertical FedGNN setting, we show that no specific optimisation method best mitigates across varying levels of data heterogeneity. Instead, they differentiate by how well they deal with an increasing number of clients and, therefore, increasingly limited data for each client. In this respect, FedProx is a clear winner in the Cora dataset experiment. The data poisoning experiments suggest the ineffectiveness of the aggregation strategies at dealing with data poisoning that does not skew the average or median to the faulty update.

References

[1] H. Brendan McMahan et al. *Communication-Efficient Learning of Deep Networks from Decentralized Data*. 2023. arXiv: 1602.05629 [cs.LG].

[2] Nicola Rieke et al. *The future of digital health with federated learning. NPJ Digital Medicine, 3, 119*. 2020.

[3] Wenqi Li et al. *Privacy-preserving Federated Brain Tumour Segmentation*. 2019. arXiv: 1910.00962 [cs.CV].

[4] Micah J. Sheller et al. *Federated learning in medicine: Facilitating multi-institutional collaborations without sharing patient data*. 2020.

[5] Zehua Sun et al. *A Survey on Federated Recommendation Systems*. 2023. arXiv: 2301.00767 [cs.IR].

[6] Tomisin Awosika, Raj Mani Shukla, and Bernardi Pranggono. *Transparency and Privacy: The Role of Explainable AI and Federated Learning in Financial Fraud Detection*. 2023. arXiv: 2312.13334 [cs.LG].

[7] Tao Liu et al. *Efficient and Secure Federated Learning for Financial Applications*. 2023. arXiv: 2303.08355 [cs.LG].

[8] Dejun Jiang et al. "Could graph neural networks learn better molecular representation for drug discovery? A comparison study of descriptor-based and graph-based models". In: *Journal of cheminformatics* 13.1 (2021), pp. 1–23.

[9] William L. Hamilton, Rex Ying, and Jure Leskovec. *Inductive Representation Learning on Large Graphs*. 2018. arXiv: 1706.02216 [cs.SI].

[10] Austin Derrow-Pinion et al. "ETA Prediction with Graph Neural Networks in Google Maps". In: *Proceedings of the 30th ACM International Conference on Information amp; Knowledge Management*. CIKM 21. ACM, 2021.

[11] Zhiyong Cui et al. *Traffic Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting*. 2019. arXiv: 1802.07007 [cs.LG].

[12] Daniel Beck, Gholamreza Haffari, and Trevor Cohn. *Graph-to-Sequence Learning using Gated Graph Neural Networks*. 2018. arXiv: 1806.09835 [cs.CL].

[13] Lingfei Wu et al. *Graph Neural Networks for Natural Language Processing: A Survey*. 2022. arXiv: 2106.06090 [cs.CL].

[14] Rui Liu et al. *Federated Graph Neural Networks: Overview, Techniques and Challenges*. 2022. arXiv: 2202.07256 [cs.DC].

[15] Chaochao Chen et al. *Vertically Federated Graph Neural Network for Privacy-Preserving Node Classification*. 2022. arXiv: 2005.11903 [cs.LG].

[16] Guangxu Mei et al. "SGNN: A Graph Neural Network Based Federated Learning Approach by Hiding Structure". In: *2019 IEEE International Conference on Big Data (Big Data)*. 2019, pp. 2560–2568.

[17] Xiang Ni et al. *A Vertical Federated Learning Framework for Graph Convolutional Network*. 2021. arXiv: 2106.11593 [cs.LG].

[18] Ke Zhang et al. *Subgraph Federated Learning with Missing Neighbor Generation*. 2021. arXiv: 2106.13430 [cs.LG].

[19] Chenhan Zhang et al. "FASTGNN: A Topological Information Protected Federated Learning Approach for Traffic Speed Forecasting". In: *IEEE Transactions on Industrial Informatics* 17.12 (2021), pp. 8464–8474.

[20] Chuan Chen et al. *FedGL: Federated Graph Learning Framework with Global Self-Supervision*. 2021. arXiv: 2105.03170 [cs.LG].

[21] Tian Li et al. *Federated Optimization in Heterogeneous Networks*. 2020. arXiv: 1812.06127 [cs.LG].

[22] Jianyu Wang et al. *Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization*. 2020. arXiv: 2007.07481 [cs.LG].

[23] Sashank Reddi et al. *Adaptive Federated Optimization*. 2021. arXiv: 2003.00295 [cs.LG].

[24] Arjun Nitin Bhagoji et al. "Analyzing Federated Learning through an Adversarial Lens". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 634–643.

[25] Subhash Sagar et al. *Poisoning Attacks and Defenses in Federated Learning: A Survey*. 2023. arXiv: 2301.05795 [cs.CR].

[26] Xiaoyu Cao and Neil Zhenqiang Gong. *MPAF: Model Poisoning Attacks to Federated Learning based on Fake Clients*. 2022. arXiv: 2203.08669 [cs.CR].

[27] Vale Tolpegin et al. "Data Poisoning Attacks Against Federated Learning Systems". In: *Computer Security – ESORICS 2020*. Ed. by Liqun Chen et al. Cham: Springer International Publishing, 2020, pp. 480–501. ISBN: 978-3-030-58951-6.

[28] Eugene Bagdasaryan et al. "How To Backdoor Federated Learning". In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, 2020, pp. 2938–2948.

[29] Dong Yin et al. *Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates*. 2021. arXiv: 1803.01498 [cs.LG].

[30] Peva Blanchard et al. "Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[31] Daniel J Beutel et al. "Flower: A Friendly Federated Learning Research Framework". In: *arXiv preprint arXiv:2007.14390* (2020).

[32] Andrew Kachites McCallum et al. "Automating the Construction of Internet Portals with Machine Learning". In: *Inf. Retr. Boston.* 3.2 (2000), pp. 127–163.

[33] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. "CiteSeer: an automatic citation indexing system". In: *Proceedings of the third ACM conference on Digital libraries*. DL '98. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 1998, pp. 89–98.

[34] Prithviraj Sen et al. "Collective classification in network data". en. In: *AI Mag.* 29.3 (2008), pp. 93–106.

[35] Thomas N Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: (2016). arXiv: 1609.02907 [cs.LG].

[36] Petar Velikovi et al. "Graph Attention Networks". 2018.

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