Federated Link Prediction on Dynamic Graphs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Link prediction on dynamic, large-scale graphs is widely used in applications such as forecasting customer visits or predicting purchases. However, graph data is often localized due to privacy and efficiency concerns. While federated learning enables collaborative training without sharing raw data, vanilla FL on full historical graphs incurs prohibitive computational costs. Training only on recent data reduces overhead but harms accuracy and introduces data imbalance across clients. We introduce FedLink, a federated graph training framework for solving link prediction tasks on dynamic graphs. By continuously training on fixed-size buffers of client data, we can significantly reduce the computation overhead compared to training on the entire historical graph, while still training a global model across regions. Experiments demonstrate that FedLink matches the accuracy of training a centralized model while requiring 3.41× less memory and running 28.9% faster compared with full-batch federated graph training.

1 Introduction

Dynamic graphs are widely used in recommendations and advertisements, where users and items form nodes and interactions form links (Figure 1 left) (Kazemi et al., 2020). This formulation enables Graph Neural Networks (GNNs) to predict user behavior (Zhang and Chen, 2018; Chen et al., 2022; Guo et al., 2023; Wang et al., 2023; Huang et al., 2024) and, by modeling data as evolving graphs, capture temporal patterns beyond static models (Pareja et al., 2020; Yu et al., 2023; Huang et al., 2023; You et al., 2022; Cong et al., 2023).

A *unified GNN model* across regions is desirable, since small regions benefit from shared data and users may move between regions (Figure 1 upper right). However, region-specific models fail to generalize, as embeddings learned in one region may not transfer well. Centralized training is limited by strict data regulations such as GDPR in Europe and PAPG in India, and by the prohibitive cost of training on billion-scale graphs (Ching et al., 2015). Federated learning (FL) addresses privacy by synchronizing local and global models (Kairouz et al., 2021; Tan et al., 2022; Ghosh et al., 2020; Deng et al., 2020; Zhou et al., 2021), but still imposes heavy overhead since each client must train on millions of nodes and edges per round.

Vanilla federated learning methods (e.g., STFL (Lou et al., 2021) and FedGraphnn (He et al., 2021)) train on the entire historical graph at each iteration. This incurs *high overhead*, since graphs with millions of users and items can contain billions of links that continually grow, requiring massive memory and training time. Using only recent snapshots (e.g., a few days of data in 4D-FED-GNN+ (Gürler and Rekik, 2022)) reduces cost but sacrifices accuracy by ignoring long-term user preferences. Other works (Wang et al., 2020; Yuan et al., 2022) train on time-series graph snapshots (e.g., hourly or daily). In practice, some regions may have only a few new links per slot (Jin et al., 2022), while others (with larger populations) generate vast interactions (Figure 1 lower right). Training only on the latest slot thus introduces *spatial and temporal heterogeneity*, which hinders FL convergence (Ye et al., 2023) by biasing the model toward data-rich clients.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

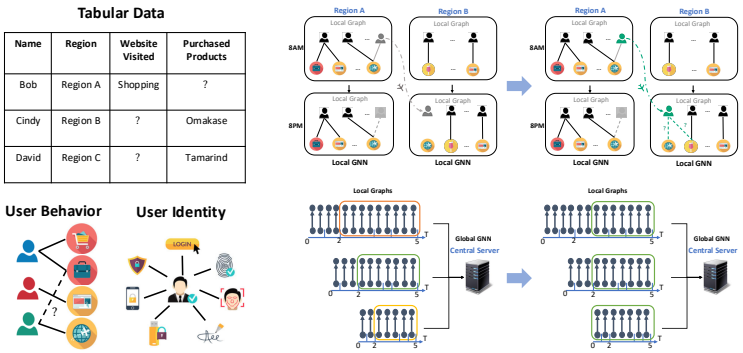

Figure 1: (Left) Use cases of federated link prediction. (Upper Right) Federated training on local dynamic graphs across regions: local graphs vary in size, clients have edges at different timestamps, and migrating users (gray→green) lose historical links for privacy, requiring new predictions. (Lower Right) Temporal imbalance across clients: uniform time-interval sampling vs. constant edge buffer.

To overcome data heterogeneity and large training overhead, we realize that ***maintaining buffers to store the same number of previously arrived edges at each client*** can both reduce computation and memory costs and ensure each client contributes equally to training (Figure 1 lower right). This enables efficient federated training and allows migrating users to transfer their embeddings across regions with minimal re-training. Our key contributions are:

- We introduce **FedLink**, a federated framework for link prediction on dynamic graphs. By training on fixed-size buffers, FedLink lowers overhead and mitigates client heterogeneity.
- We provide **theoretical analysis and empirical validation** of buffer size, showing tradeoffs between staleness, dataset size, memory, and training time.
- **Experiments** across regions show FedLink reduces GPU memory (up to 3.41×) and training time (28.9%) compared to full-batch FL, while matching prediction accuracy.

Section 2 reviews related work; Section 3 introduces FedLink with theoretical and empirical validation; Section 4 presents distributed experiments; and Appendix A concludes.

2 Related Work

Graph neural networks learn representations of graph-structured data (Bronstein et al., 2017). Models such as GCN (Kipf and Welling, 2016), GraphSage (Hamilton et al., 2017), and GAT (Veličković et al., 2017) achieve strong performance on tasks like node classification. Dynamic GNNs extend these methods by incorporating temporal information through recurrent structures (Chen et al., 2022; Pareja et al., 2020; Yu et al., 2023) or specialized training (Huang et al., 2023; You et al., 2022; Cong et al., 2023), with link prediction as a central task (Zhang and Chen, 2018).

Federated learning (Kairouz et al., 2021; Tan et al., 2022; Ghosh et al., 2020; Deng et al., 2020; Zhou et al., 2021) enables distributed training with privacy preservation. Several methods extend FL to GNNs on static graphs (Liu et al., 2024; Wang et al., 2022), including GCFL (Xie et al., 2021), FedSage+ (Zhang et al., 2021), FedGCN (Yao et al., 2024a), and FedPub (Baek et al., 2023), supported by libraries like FedGraph (Yao et al., 2024b). For **dynamic graphs**, STFL (Lou et al., 2021) applies spatio-temporal models, Feddy (Jiang et al., 2022) uses dynamic embeddings, and 4D-FED-GNN+ (Gürler and Rekik, 2022) handles evolution with missing time points. These methods, however, overlook client and temporal imbalance and often require training on full historical graphs, which is costly and may include outdated links. To our knowledge, *FedLink is the first federated link prediction method on dynamic graphs* that directly addresses these challenges, as shown in Section 4.

3 FedLink

We first formalize the federated link prediction problem and introduce the FedLink algorithm. We then present a theoretical rationale and empirical validation of FedLink's buffer-based design.

3.1 Federated Link Prediction

Federated graph setup. We consider K clients (regions) and a central server. Each client k has users $\{i = 1, \ldots, I_k\}$, items $\{j = 1, \ldots, J_k\}$, and a local graph $\mathcal{G}_{k,t}$ built from user–item interactions $e_{k,t}^{(i,j)}$ up to time t. Users may move across regions, but items remain local (e.g., restaurants). The graph $\mathcal{G}_{k,t}$ grows as edges accumulate.

Link prediction formulation. At time T, client k has user set $\mathcal{I}_k$, item set $\mathcal{J}_k$, and local graph $\mathcal{G}_{k,T}$. Training initializes global embeddings $\boldsymbol{I} \in \mathbb{R}^{N \times d}$ and $\boldsymbol{J} \in \mathbb{R}^{M \times d}$. A GNN with parameters $\boldsymbol{w}$ learns user/item representations $\boldsymbol{\theta}$, and a predictor $\boldsymbol{\phi}$ estimates the probability of future edges:

$$\mathbb{P}\Big(e_{k,T'}^{(i,j)} \in \mathcal{G}_{k,T'} \mid \boldsymbol{\theta}, \boldsymbol{\phi}, \mathcal{I}_k, \mathcal{J}_k, \mathcal{G}_{k,T}\Big). \tag{1}$$

We define $\mathbb{P}(e_{k,t}^{(i,j)}) = \boldsymbol{\phi}(s(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j))$, where $s(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \frac{\boldsymbol{\theta}_i \cdot \boldsymbol{\theta}_j}{\|\boldsymbol{\theta}_i\| \|\boldsymbol{\theta}_j\|}$ is the cosine similarity (Zhang and Chen, 2018; Kipf and Welling, 2016; Yao et al., 2024a).

Federated learning for link prediction. Training proceeds in rounds $r = 1, \ldots, R$. At round r, each client k trains on $\mathcal{G}_{k,t}$ with L local SGD steps and uploads updated model $\boldsymbol{w}_k^{(r)}$ and embeddings $(\boldsymbol{I}_k^{(r)}, \boldsymbol{J}_k^{(r)})$. The server averages updates to obtain the global model $(\boldsymbol{w}^{(r)}, \boldsymbol{I}^{(r)}, \boldsymbol{J}^{(r)})$. After R rounds, each client predicts future links with its trained model and local graph.

Federated learning for link prediction. Given the link prediction model above, we next explain how federated learning can be used to train this model. As usual in federated learning, training proceeds in rounds. Without loss of generality, suppose that training takes place at rounds $r = 1, 2, \ldots, R$; note that since we model edge arrivals as a continuous-time process, new edges may arrive in between the training rounds. At a specific round r that takes place at time t, each client k trains its local link prediction model on the local graph $\mathcal{G}_{k,t}$ with L local stochastic gradient descent steps. Each client k then sends its updated GNN model $\boldsymbol{w}_k^{(r)}$ and embedding layers $(\boldsymbol{I}_k^{(r)}, \boldsymbol{J}_k^{(r)})$ to the global server, which averages the received local models to update the global model $\boldsymbol{w}^{(r)}$ and global embedding layers $(\boldsymbol{I}^{(r)}, \boldsymbol{J}^{(r)})$. Once R rounds of training have taken place, where R can be chosen to ensure convergence, each client k uses the trained model and its local graph $\mathcal{G}_{k,t}$ to predict the future links.

3.2 FedLink Algorithm

A key challenge in FL is the **growing computation overhead** as local graphs $\mathcal{G}_{k,t}$ expand, with millions of users and items leading to heavy memory and runtime costs. Beyond reducing overhead, FedLink addresses three issues: **Client Heterogeneity:** Different regions generate vastly different numbers of interactions. **Temporal Heterogeneity:** Link arrivals vary over time and across regions (e.g., by time zones). **Cross-client Users:** Users may move between regions, requiring cross-client adaptation. FedLink tackles these challenges with two features: a *constant edge buffer* and *user embedding sharing*, detailed below.

3.2.1 Constant Edge Buffer

Since links arrive continuously, triggering a training round for every new link causes prohibitive communication overhead, while fixed-time rounds (e.g., daily) yield imbalance across clients with different link rates (Figure 1, bottom middle).

To address this, we partition the link stream into *buffers* of a constant number of edges C. For a temporal network $\mathcal{G}$ with edge stream $\{e^1, \ldots, e^m\}$, buffers are defined as

$$\mathcal{G}^s = \{e^i \mid C(s-1) < i \leq Cs\},$$

with a FIFO policy replacing old edges once capacity is reached (Figure 1, bottom right).

In FedLink (Algorithm 1), each client receives the global model $\boldsymbol{W}^{(r)}$, performs L local gradient steps using one buffer per step, and returns updates for server aggregation. Unlike training on the full graph, clients only store current buffer contents, greatly reducing memory and computation. Sampling from past buffers also mitigates overfitting to the latest data, maintaining accuracy comparable to historical training. Finally, buffers are updated solely by link age: even if users move across regions, their past edges remain until expired, avoiding premature deletion.

Algorithm 1 `FedLink`

Model parameters are represented by $\boldsymbol{W} = (\boldsymbol{w}, \boldsymbol{I}, \boldsymbol{J})$.
Each client k maintains buffer graphs $\{\mathcal{G}_k^1, \mathcal{G}_k^2, \ldots, \mathcal{G}_k^S\}$, each with constant number of edges C.
for *round* $r = 1, \ldots, R$ **do**
  for *each client* $k \in [K]$ **do in parallel**
    Receive $\boldsymbol{W}^{(r)}$
    Set $\boldsymbol{W}_k^{(r,1)} = \boldsymbol{W}^{(r)}$
    for *local step* $l = 1, \ldots, L$ **do**
      Choose a buffer $\mathcal{G}_k^s \in \{\mathcal{G}_k^1, \mathcal{G}_k^2, \ldots, \mathcal{G}_k^S\}$
      Set $\boldsymbol{g}_{\boldsymbol{W}_k}^{(r,l)} = \nabla f_k(\boldsymbol{W}_k^{(r,l)}; \mathcal{G}_k^s)$
      $\boldsymbol{W}_k^{(r,l+1)} = \boldsymbol{W}_k^{(r,l)} - \eta\, \boldsymbol{g}_{\boldsymbol{W}_k}^{(r,l)}$ `// Update Parameters`
    end
    $\boldsymbol{\Delta}_{\boldsymbol{W}_k}^{(r,L)} = \boldsymbol{W}_k^{(r,L+1)} - \boldsymbol{W}_k^{(r,1)}$
    Send $\boldsymbol{\Delta}_{\boldsymbol{W}_k}^{(r,L)}$ to the server
  end
  `// Server Operations`
  $\boldsymbol{\Delta}_{\boldsymbol{W}}^{(r)} = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{\Delta}_{\boldsymbol{W}_k}^{(r,L)}$ `// Difference Aggregation`
  $\boldsymbol{W}^{(r+1)} = \boldsymbol{W}^{(r)} + \boldsymbol{\Delta}_{\boldsymbol{W}}^{(r)}$ and broadcast to local clients `// Update Global Models`
end
`// Perform Link Prediction with sharing of user embedding` $\boldsymbol{I}$
for *each client* $k \in [K]$ **do in parallel**
  Predict future links using (1) based on $\boldsymbol{W}^{R+1}, \mathcal{G}_{k,T}$
end

3.2.2 Cross-Client User Embeddings

We next outline how FedLink generates link predictions when users move across clients. We take advantage of the fact that the link prediction mainly relies on the user and item embeddings $(\boldsymbol{I}, \boldsymbol{J})$, which is shared across clients. In order to generate link predictions for a user newly arrived at a client k', we can *reuse the embedding* $\boldsymbol{I}$ *for this user that was learned at the user's previous client* k. Since the items available at client k' may differ from those at client k, users may receive less accurate predictions as their embeddings were not trained to predict link formation for client k''s items. However, these predictions should be more accurate than if client k' learned the user's embedding from scratch. We empirically validate this intuition in Section 4's evaluation.

3.3 Theoretical Analysis of FedLink

FedLink trains GNNs on buffered past data, so the choice of *buffer size* strongly affects convergence. To study this, we use the dynamic stochastic block model (DSBM) (Abbe, 2018; Keriven and Vaiter, 2022) (Appendix C) to approximate how buffer size impacts accuracy.

3.3.1 Buffer Error

Using Equation (5), we derive the dependence on buffer size. For client k with link arrival rate λ_k, a buffer of size C covers ϵ/λ_k time and yields a local error:

$$E_{buf}(C, \lambda_k) = \int_0^{C/\lambda_k} E(\tau) d\tau = 2nC/\lambda_k - \frac{(1-\varepsilon)^{\eta C/\lambda_k} - 1}{\eta \log(1-\varepsilon)}. \tag{2}$$

The global error is $E_{\text{classification}} = \frac{1}{K}\sum_k^K E_{buf}(C, \lambda_k)$. The first term grows linearly with C (staleness), while the second decays exponentially and is bounded. Thus, large buffers increase stale data but also improve convergence (Ye et al., 2023), since consistent dataset sizes across clients reduce gradient variance. Buffer size therefore balances staleness, convergence, and resource costs.

3.3.2 Buffer Selection

We empirically test buffer sizes (10k–1M for full data, 10k–500k for 50% downsampled) across regions ranging from under 100k to nearly 2M check-ins (US: 1,990,327). Figure 2 shows the expected trade-off: small buffers fail to capture history in large regions, while oversized buffers exceed data in small ones, producing linear performance drops consistent with Equation (2). The concave AUC trend validates our theory, and we adopt these optimal ranges in experiments.

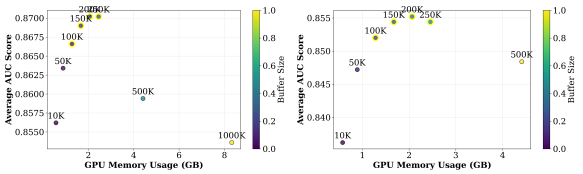

Figure 2: Impact of buffer size on AUC scores and GPU usage for (left) original data and (right) 50% downsampled data across regions. Optimal buffer sizes balancing AUC and GPU usage are highlighted in yellow.

4 Experiments

In experiments, we address the following research questions: **RQ1. System Efficiency:** Does FedLink reduce GPU memory usage and training time compared to baselines? **RQ2. Model Accuracy:** Does the buffer method maintain accuracy comparable to traditional FL while handling regional heterogeneity? **RQ3. Buffer & Data Efficiency:** How do embedding sharing, buffer design, and dataset size affect performance and efficiency?

4.1 Datasets and Baselines

Datasets: **Foursquare** (Yang et al., 2016), treating each country as a client with user–venue edges (10 representative countries, Appendix D.2), and **TGBL-Wiki** (Huang et al., 2023; Kumar et al., 2019) from the Temporal Graph Benchmark, containing temporal user–page interactions (Appendix D.6). Baselines: **Local** (Hamilton et al., 2017): GNN on local graphs only. **STFL** (Lou et al., 2021): Spatio-temporal FL with static GNNs. **4D-FED-GNN+** (Gürler and Rekik, 2022): FL on daily edge snapshots. **FedDGL** (Xie et al., 2024): Dynamic FL with distillation and prototype regularization. **Feddy** (Jiang et al., 2022): Combines spatial and temporal aggregation. **FedLink** (ours): Edge buffers and embedding transfer. **FedLink-NoEmb**: Without embedding transfer. **FedLink-Local**: Local training only, no aggregation. **FedLink-MiniBatch**: Mini-batch sampling with buffer-size batches.

4.2 Experiment Results

RQ1: System Evaluation: We assess FedLink's efficiency by measuring training time, GPU memory, and AUC (Table 1). With a buffer size of 200k, FedLink matches or outperforms 4D-FED-GNN+ and FedDGL in speed, while using at least 3.41× less GPU memory than full-batch methods (STFL, Feddy), all with competitive AUC. Detailed per-country results are in Table C.1.

RQ2: Model Accuracy: Table 1 shows FedLink attains AUC scores comparable to the best methods (STFL, Local, FedDGL), consistently ranking among the top two. As illustrated in Figure 3, FedLink achieves high accuracy with far lower GPU and training costs, unlike 4D-FED-GNN+, which is efficient but much less accurate. Thus, FedLink uniquely balances efficiency and predictive performance.

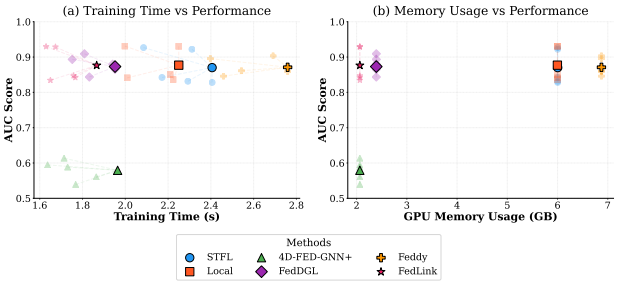

Figure 3: Comparison of methods: (a) training time vs. AUC, and (b) GPU memory vs. AUC. FedLink achieves the best balance of accuracy, time, and memory.

RQ 3: Ablation Study: In order to evaluate the contribution of each component within FedLink, we conduct an ablation study comparing four variants of FedLink. FedLink constantly achieves the highest AUC scores among all variants, demonstrating the effectiveness of both federated learning setting and FIFO buffer mechanisms, as shown in Table 2. The results of isolating only traveled users for different FedLink methods are in Table C.4.

Table 1: Performance comparison across six experimental settings. Measurements include training time (seconds), GPU memory usage (GB), and AUC scores. Results are averaged over 10 runs. **Bold** indicates best results, ***bold italics*** indicate second-best.

Exp 1 (US, BR, ID, TR, JP)				**Exp 2** (MX, PH, ES, GB, IT)			
Methods	Time(s)	GPU(GB)	AUC	Methods	Time(s)	GPU(GB)	AUC
STFL	2.406	6.003	0.870	STFL	2.082	1.220	**0.848**
Local	2.250	5.997	**0.877**	Local	***1.867***	1.220	0.847
4D-FED-GNN+	1.964	***2.065***	0.579	4D-FED-GNN+	**1.857**	**0.894**	0.567
FedDGL	***1.932***	**1.892**	0.873	FedDGL	1.950	0.933	0.832
Feddy	2.759	7.574	0.871	Feddy	2.389	2.386	0.841
FedLink	**1.866**	***2.065***	***0.876***	FedLink	1.997	***0.894***	***0.848***
Exp 3 (US, JP, BR, MX, ES)				**Exp 4** (DE, NL, KR, FR, CA)			
Methods	Time(s)	GPU(GB)	AUC	Methods	Time(s)	GPU(GB)	AUC
STFL	2.058	5.832	0.869	STFL	1.752	0.886	0.861
Local	1.889	5.812	**0.876**	Local	1.624	0.869	0.860
4D-FED-GNN+	**1.649**	***1.789***	0.587	4D-FED-GNN+	***1.484***	***0.678***	0.598
FedDGL	1.897	**1.773**	***0.867***	FedDGL	1.519	0.790	***0.862***
Feddy	2.254	6.184	0.871	Feddy	2.081	1.214	0.858
FedLink	***1.782***	***1.789***	***0.871***	FedLink	**1.463**	**0.676**	**0.864**
Exp 5 (TGBL-Wiki, Early Period)				**Exp 6** (TGBL-Wiki, Later Period)			
Methods	Time(s)	GPU(MB)	AUC	Methods	Time(s)	GPU(MB)	AUC
STFL	0.735	0.589	0.868	STFL	0.762	0.618	***0.859***
Local	0.711	0.589	0.866	Local	0.738	0.618	0.852
4D-FED-GNN+	**0.582**	**0.303**	0.824	4D-FED-GNN+	**0.607**	**0.319**	0.806
FedDGL	0.713	0.706	***0.869***	FedDGL	0.740	0.713	0.855
Feddy	1.185	0.943	0.864	Feddy	1.234	0.996	0.851
FedLink	***0.617***	***0.365***	**0.875**	FedLink (Buffer)	***0.649***	***0.371***	**0.867**

Table 2: Ablation study by removing the FL part and the cross-client embedding part. Federated aggregation significantly contributes to FedLink's performance.

Buffer Size = 200,000					
Methods	US	BR	ID	TR	JP
FedLink (FL+Buffer)	**0.837±0.003**	0.930±0.004	**0.834±0.004**	**0.929±0.003**	**0.842±0.005**
FedLink-Local	0.828±0.003	0.931±0.004	0.827±0.003	0.923±0.004	0.840±0.005
FedLink-NoEmb	0.832±0.003	**0.931±0.006**	0.829±0.006	0.931±0.005	0.836±0.007
FedLink-MiniBatch	0.823±0.005	0.926±0.004	0.825±0.005	0.921±0.005	0.834±0.006
Buffer Size = 300,000					
Methods	US	BR	ID	TR	JP
FedLink (FL+Buffer)	**0.848±0.003**	**0.940±0.004**	**0.906±0.004**	**0.966±0.003**	**0.950±0.005**
FedLink-Local	0.832±0.003	0.926±0.006	0.890±0.006	0.963±0.005	0.946±0.007
FedLink-NoEmb	0.847±0.003	0.938±0.004	0.904±0.004	0.965±0.003	0.946±0.006
FedLink-MiniBatch	0.831±0.005	0.934±0.006	0.893±0.006	0.957±0.004	0.941±0.007

To evaluate FedLink's performance under different data sizes, we conduct extensive downsampling experiments. Our experimental results demonstrate FedLink's resilience to data reduction, maintaining among highest accuracy with 50% downsampling of the training data shown in Figure 4. We also performed downsampling experiments for 50%, 25%, and 2% data size for different country combinations, which can be found in Table C.3.

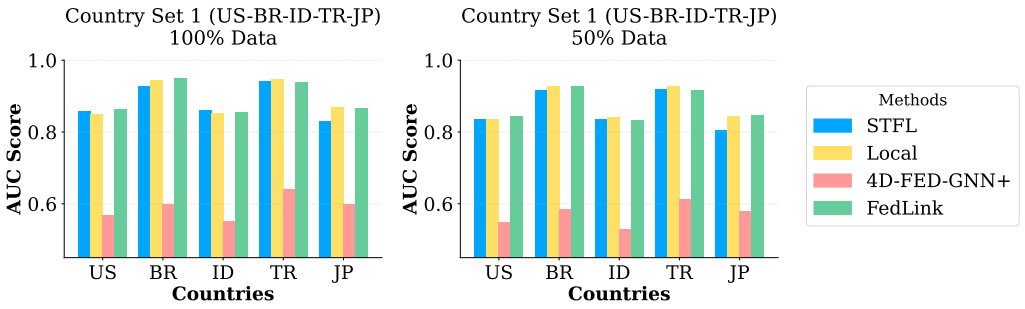

Figure 4: Performance comparison under data downsampling (100% vs 50%).

References

Emmanuel Abbe. 2018. Community detection and stochastic block models: recent developments. *Journal of Machine Learning Research* 18, 177 (2018), 1–86.

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

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.

Jinyin Chen, Xueke Wang, and Xuanheng Xu. 2022. GC-LSTM: Graph convolution embedded LSTM for dynamic network link prediction. *Applied Intelligence* (2022), 1–16.

Avery Ching, Sergey Edunov, Maja Kabiljo, Dionysios Logothetis, and Sambavi Muthukrishnan. 2015. One trillion edges: Graph processing at facebook-scale. *Proceedings of the VLDB Endowment* 8, 12 (2015), 1804–1815.

Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. 2023. Do we really need complicated model architectures for temporal networks? *arXiv preprint arXiv:2302.11636* (2023).

Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. 2020. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461* (2020).

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems* 33 (2020), 19586–19597.

Zhichun Guo, William Shiao, Shichang Zhang, Yozen Liu, Nitesh V Chawla, Neil Shah, and Tong Zhao. 2023. Linkless link prediction via relational distillation. In *International Conference on Machine Learning*. PMLR, 12012–12033.

Zeynep Gürler and Islem Rekik. 2022. Federated Brain Graph Evolution Prediction using Decentralized Connectivity Datasets with Temporally-varying Acquisitions. *IEEE Transactions on Medical Imaging* (2022).

William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

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

Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. 2023. Temporal graph benchmark for machine learning on temporal graphs. *Advances in Neural Information Processing Systems* 36 (2023), 2056–2073.

Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. 2024. Temporal graph benchmark for machine learning on temporal graphs. *Advances in Neural Information Processing Systems* 36 (2024).

Meng Jiang, Taeho Jung, Ryan Karl, and Tong Zhao. 2022. Federated Dynamic Graph Neural Networks with Secure Aggregation for Video-based Distributed Surveillance. *ACM Transactions on Intelligent Systems and Technology (TIST)* 13, 4 (2022), 1–23.

Di Jin, Sungchul Kim, Ryan A Rossi, and Danai Koutra. 2022. On Generalizing Static Node Embedding to Dynamic Settings. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 410–420.

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

Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. 2020. Representation learning for dynamic graphs: A survey. *The Journal of Machine Learning Research* 21, 1 (2020), 2648–2720.

Nicolas Keriven and Samuel Vaiter. 2022. Sparse and smooth: improved guarantees for spectral clustering in the dynamic stochastic block model. *Electronic Journal of Statistics* 16, 1 (2022), 1330–1366.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1269–1278.

Xiaoyu Li and Francesco Orabona. 2019. On the convergence of stochastic gradient descent with adaptive stepsizes. In *The 22nd international conference on artificial intelligence and statistics*. PMLR, 983–992.

Rui Liu, Pengwei Xing, Zichao Deng, Anran Li, Cuntai Guan, and Han Yu. 2024. Federated graph neural networks: Overview, techniques, and challenges. *IEEE Transactions on Neural Networks and Learning Systems* (2024).

Guannan Lou, Yuze Liu, Tiehua Zhang, and Xi Zheng. 2021. STFL: A temporal-spatial federated learning framework for graph neural networks. *arXiv preprint arXiv:2111.06750* (2021).

Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 5363–5370.

Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. 2022. Towards personalized federated learning. *IEEE transactions on neural networks and learning systems* 34, 12 (2022), 9587–9603.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

Binghui Wang, Ang Li, Meng Pang, Hai Li, and Yiran Chen. 2022. Graphfl: A federated learning framework for semi-supervised node classification on graphs. In *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 498–507.

Huan Wang, Ziwen Cui, Ruigang Liu, Lei Fang, and Ying Sha. 2023. A multi-type transferable method for missing link prediction in heterogeneous social networks. *IEEE Transactions on Knowledge and Data Engineering* 35, 11 (2023), 10981–10991.

Junshan Wang, Guojie Song, Yi Wu, and Liang Wang. 2020. Streaming graph neural networks via continual learning. In *Proceedings of the 29th ACM international conference on information & knowledge management*. 1515–1524.

Han Xie, Jing Ma, Li Xiong, and Carl Yang. 2021. Federated graph classification over non-iid graphs. *Advances in neural information processing systems* 34 (2021), 18839–18852.

Zaipeng Xie, Li Likun, Xiangbin Chen, Hao Yu, and Qian Huang. 2024. FedDGL: Federated Dynamic Graph Learning for Temporal Evolution and Data Heterogeneity. In *The 16th Asian Conference on Machine Learning (Conference Track)*.

Dingqi Yang, Daqing Zhang, and Bingqing Qu. 2016. Participatory cultural mapping based on collective behavior data in location-based social networks. *ACM Transactions on Intelligent Systems and Technology (TIST)* 7, 3 (2016), 1–23.

Yuhang Yao, Weizhao Jin, Srivatsan Ravi, and Carlee Joe-Wong. 2024a. FedGCN: Convergence-communication tradeoffs in federated training of graph convolutional networks. *Advances in neural information processing systems* 36 (2024).

Yuhang Yao, Yuan Li, Xinyi Fan, Junhao Li, Kay Liu, Weizhao Jin, Srivatsan Ravi, Philip S Yu, and Carlee Joe-Wong. 2024b. FedGraph: A Research Library and Benchmark for Federated Graph Learning. *arXiv preprint arXiv:2410.06340* (2024).

Mang Ye, Xiuwen Fang, Bo Du, Pong C Yuen, and Dacheng Tao. 2023. Heterogeneous federated learning: State-of-the-art and research challenges. *Comput. Surveys* 56, 3 (2023), 1–44.

Jiaxuan You, Tianyu Du, and Jure Leskovec. 2022. ROLAND: graph learning framework for dynamic graphs. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 2358–2366.

Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. 2023. Towards better dynamic graph learning: New architecture and unified library. *Advances in Neural Information Processing Systems* 36 (2023), 67686–67700.

Xiaoming Yuan, Jiahui Chen, Jiayu Yang, Ning Zhang, Tingting Yang, Tao Han, and Amir Taherkordi. 2022. FedSTN: Graph Representation Driven Federated Learning for Edge Computing Enabled Urban Traffic Flow Prediction. *IEEE Transactions on Intelligent Transportation Systems* (2022).

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

Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *Advances in neural information processing systems* 31 (2018).

Zirui Zhou, Lingyang Chu, Changxin Liu, Lanjun Wang, Jian Pei, and Yong Zhang. 2021. Towards fair federated learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 4100–4101.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: As in abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Discussed in conclusion.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Provided in appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: As in Appendix. Code and data will also be open-source.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The workshop does not have a code submission portal but can be provided.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: In main paper and appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: As in Table 2.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: As in appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: Follows the code of ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: As in conclusion

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Cited papers.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Documented in a github repo.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Conclusion

In this paper, we propose FedLink, a federated link prediction algorithm for dynamic graphs. FedLink is motivated by the problem of recommending items, e.g., restaurants, to users in multiple regions. While it is desirable to train a model across regions to take full advantage of all available data, privacy constraints, and computational overhead may prohibit centralized training of a dynamic GNN model across all regions. FedLink addresses the challenges of computational overhead and privacy concerns in situations where graph data is localized, such as recommending restaurants to users in multiple countries. This approach has the added benefit of accommodating users who move across countries, by allowing them to simply share the user embedding across countries. Moreover, FedLink maintains edge buffers of fixed size at each client, thus alleviating the effects of temporal and inter-client heterogeneity in link arrivals over time. We show that FedLink significantly reduces memory requirements and improves training speed while matching the accuracy of centralized training. Edge buffers of fixed size may not fully utilize the temporal information. Future work includes dynamic buffer and extending FedLink to other dynamic graph applications and generalizing the standard federated learning convergence analysis to address the unique challenges of dynamic graph settings. Our methodology also has the potential to extend to time-series data analysis, node classification, and graph classification[1].

B Background

B.1 Graph Convolutional Network

A multi-layer Graph Convolutional Network (GCN) Kipf and Welling (2016) with row normalization has the layer-wise propagation rule

$$H^{(l+1)} = \phi(\widetilde{D}^{-1}\widetilde{A}H^{(l)}W^{(l)}), \tag{3}$$

where $\widetilde{A} = A + I_N$, I_N is the identity matrix, $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$ and $W^{(l)}$ is a layer-specific trainable weight matrix. The activation function is ϕ, typically ReLU (rectified linear units), with a softmax in the last layer for node classification. The node embedding matrix in the l-th layer is $H^{(l)} \in \mathbb{R}^{N\times D}$, which contains high-level representations of the graph nodes transformed from the initial features; $H^{(0)} = X$.

C Theoretical Analysis

C.1 Dynamic Stochastic Block Model

For positive integers K and n, a probability vector $p \in [0,1]^K$, and a symmetric connectivity matrix $B \in [0,1]^{K\times K}$, we define a static SBM as a random graph with n nodes split into K classes. The goal of a prediction method for the SBM is to correctly divide nodes into their corresponding classes, based on the graph structure. Each node is independently and randomly assigned a class in $\{1, ..., K\}$ according to the distribution p. Undirected edges are independently created between any pair of nodes in classes i and j with probability B_{ij}, which equals α if $i = j$ (i and j are in the same class) and $\mu\alpha$ if $i \neq j$ (i and j are in different classes), where $\alpha \in (0,1)$ and $\mu \in (0,1)$ are given parameters.

We consider a set of discrete time steps $t = 1, 2, \ldots, T$. At each time step t, the Dynamic SBM generates new intra- and inter-class edges according to the probabilities α and $\mu\alpha$ as defined for the SBM above. All edges persist over time. We assume a constant number of nodes n, number of classes K, and connectivity matrix B. Let $Y_t \in \{0,1\}^{n\times K}$ denote the matrix representing the nodes' class memberships at each time t, where $Y_{ik} = 1$ indicates that node i belongs to the k-th class, and is 0 otherwise. We model changes in nodes' class memberships as a Markov process with a constant transition probability matrix $H \in [0,1]^{K\times K}$. Let $\varepsilon \in (0,1)$ denote the probability a node changes its membership. At each time step, node v_i in class j changes its membership to class k with the

[1] Code and data will be made publicly available upon acceptance.

following probability (independently from other nodes):

$$H_{j,k} = \mathbb{P}\left[Y_{ik}^{t} = 1 | Y_{ij}^{t-1} = 1\right] = \begin{cases} 1-\varepsilon, & j = k \\ \dfrac{\varepsilon}{K-1}, & j \neq k, \end{cases}$$

While ε may vary across classes j in practice, for simplicity we suppose it is the same for each class.

We suppose that our learning task is to classify the nodes of the graph, i.e., to group nodes together so as to recover the membership matrix Y up to column permutation at each time t. We expect link prediction to give similar convergence results, but as the analysis is more involved we present the node classification analysis for simplicity. We thus evaluate estimates $\hat{Y}$ of the membership matrix by defining the *relative error* of a classification estimate $\hat{Y}$ as

$$E(\hat{Y}, Y) = \min_{\pi \in \mathcal{P}} \|\hat{Y}\pi - Y\|_0, \tag{4}$$

where $\mathcal{P}$ is the set of all $K \times K$ permutation matrices and $\|.\|_0$ counts the number of non-zero elements of a matrix.

C.2 Class Behavior Over Time

We observe the behavior of class evolution over time by using the relative error function in (4) to characterize the change in classification over time, i.e. $E(Y_{t-\tau}, Y_t)$. Without loss of generality, we remove the permutation and keep class indices for columns of Y constant for all membership matrices. Since node transitions are independent, the probability matrix $\mathbb{E}[Y_t|Y_{t-1}] = Y_{t-1}H^T$ allows us to find the expectation of the relative error between adjacent time steps. This is obtained by modeling the error as n individual Markov chains between correct and incorrect classifications for each node. The probability of incorrectly predicting a node's class using the previous time step's information is ε (the probability of class shifting between time steps). Therefore, $\mathbb{E}[E(Y_{t-1}, Y_t)] = 2\|Y_{t-1}\|_0\varepsilon = 2\|Y_t\|_0\varepsilon = 2n\varepsilon$, since every classification mistake increments the error metric by 2. For further time steps, due to the penalty function used in $E(\cdot,\cdot)$, we construct a two-state Markov chain for each user, where the states denote whether the user is in the same or different class as the current time. Then as $t \to \infty$, the system reaches a stationary distribution and $\mathbb{E}[E(Y_{t-\tau}, Y_t)] = 2n\frac{K-1}{K} \approx 2n$ for large K. We will approximate the error over time by the following continuous function:

$$E(t) = 2n - 2n(1-\varepsilon)^{\eta t} \tag{5}$$

with an arbitrary convergence factor η (Li and Orabona, 2019).

D Additional Experiments

D.1 Experiment Settings

Building upon our baseline methods, we evaluate FedLink's link prediction performance in our four subsets of countries in the Foursquare dataset. Based on the buffer size analysis in Section 3.3.2, we employ buffer sizes of 200,000 and 300,000, which demonstrated an optimal range for computational efficiency and model performance. In order to capture patterns of check-in behaviors and sufficient user travel across countries, we analyze the data over a 30-day period. The training settings for each model are as follows: **Local Training (Local and FedLink-Local)**: Both methods perform 60 local training iterations. Local uses the complete dataset, while FedLink-Local uses a single buffer per iteration. **Federated Training (STFL, 4D-Fed-GNN+,FedDGL, and Feddy)**: All methods perform 20 global training rounds with 3 local iterations per client. STFL, FedDGL and Feddy processes full historical data, while 4D-FED-GNN+ uses single-day data, representing scenarios with limited computational capacity. FedDGL selectively processes 10% of nodes as sensitive information for temporal knowledge distillation. **Federated Training with Buffer (FedLink, FedLink-NoEmb, and FedLink-MiniBatch)**: All methods perform 20 global training rounds with 3 local iterations per client, processing one buffer per iteration. FedLink shares user embeddings across clients for traveled users, while FedLink-NoEmb operates without this cross-client information sharing. FedLink-MiniBatch uses random sampling within buffers instead of FIFO sequential processing.

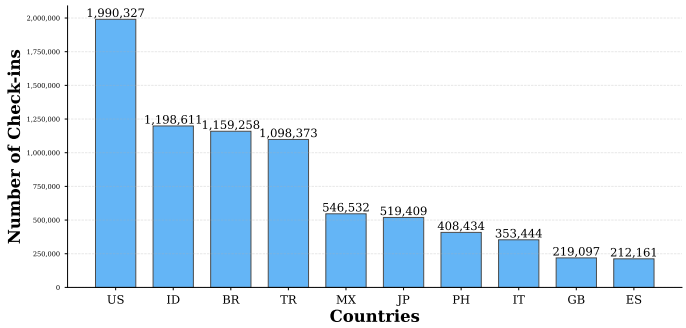

Figure C.2: Distribution of check-ins across selected 10 countries in the Foursquare dataset.

D.2 Foursquare Dataset

We present the Foursquare dataset with the following statistics: Check-ins: 7,705,646 check-ins made by 40,484 users across 1,273,946 venues. Population Distribution: Selected countries show significant variation in user population sizes, reflecting real-world demographic differences, shown in Figure C.1. Check-in Distribution: The check-in volumes also vary across countries, with the United States (US) having 1,990,327 check-ins, while Spain (ES) having 212,161 check-ins for example, shown in Figure C.2. Traveled user percentage: Approximately 1.39% of users have check-ins across multiple countries, which we refer to as traveled users. We organize our experiments into four sets according to different country sizes as follows:
Large Countries (EXP 1): Top five countries with the highest check-in volumes: United States (US), Brazil (BR), Indonesia (ID), Turkey (TR), and Japan (JP) ;
Midsized countries (EXP 2): Five countries with relatively limited data: Mexico (MX), Philippines (PH), Spain(ES), UK(GB), and Italy(IT);
Combinations (EXPs 3): Combinations of large and small countries (US, JP, BR, MX, ES) to assess FedLink's performance under client data heterogeneity;
Small Countries (EXP 4) Five small countries with size around 5% of the size of US: Germany(DE), Netherlands(NL), South Korea(KR), France(FR), and Canada(CA).

We present Fig C.1 to show the population distribution across the ten countries selected for our experiments. This population heterogeneity allow us to assess our model's ability to handle different levels of data density and user activity patterns.

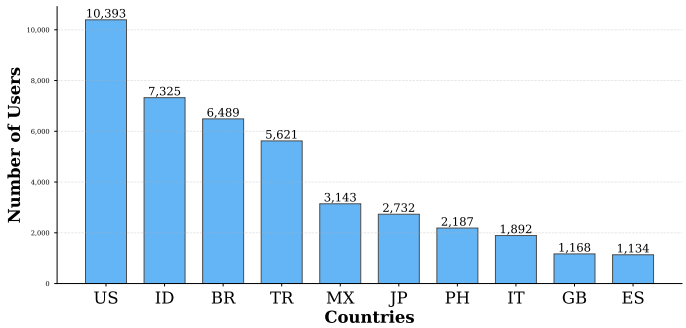

Figure C.1: Distribution of user population across selected 10 countries in the Foursquare dataset.

D.3 Experimental Results

We present detailed performance metrics for different country combinations in Table C.1, showing training time for each country, total completion time, and GPU memory usage during training. The total training time per round is determined by the slowest client, as federated learning requires all clients to complete their local training before proceeding with global model updates. These detailed

Table C.1: Training Time of FedLink with buffer size 200,000 on clients for 1200 Iterations. Each result represents the total training time, averaged over 10 runs. FedLink consistently trains faster and has lower GPU memory usage, than all other algorithms except 4D-FED-GNN+ which has a much lower AUC.

	US	BR	ID	TR	JP	Total Time (s)	Training GPU (GB)
STFL	2.172	2.085	2.293	2.312	2.406	2.406	5.997
Local	2.009	1.998	2.223	2.250	2.210	2.250	5.997
4D-FED-GNN+	1.864	1.637	1.768	1.714	1.730	***1.864***	**2.065**
FedLink	1.758	1.630	1.650	1.673	1.766	**1.766**	**2.065**
	MX	PH	ES	GB	IT	Total Time (s)	Training GPU (GB)
STFL	1.688	2.082	1.755	1.624	1.551	2.082	1.220
Local	1.540	1.867	1.580	1.577	1.549	***1.867***	1.220
4D-FED-GNN+	1.493	1.857	1.488	1.481	1.480	**1.857**	**0.894**
FedLink	1.583	1.996	1.602	1.689	1.690	1.996	***0.894***
	US	BR	JP	MX	ES	Total Time (s)	Training GPU (GB)
STFL	2.058	1.881	1.711	1.866	1.994	2.058	5.832
Local	1.798	1.837	1.650	1.889	1.809	1.889	5.830
4D-FED-GNN+	1.606	1.601	1.594	1.649	1.583	**1.649**	***1.789***
FedLink	1.648	1.630	1.630	1.782	1.781	***1.782***	**1.789**
	US	IT	GB	TR	ES	Total Time(s)	Training GPU(GB)
STFL	2.015	1.894	1.723	1.879	1.982	2.015	4.449
Local	1.806	1.860	1.662	1.867	1.814	1.860	4.447
4D-FED-GNN+	1.650	1.614	1.607	1.643	1.596	**1.650**	***1.302***
FedLink	1.554	1.627	1.624	1.632	1.639	**1.639**	**1.301**
	DE	NL	KR	FR	CA	Total Time(s)	Training GPU(GB)
STFL	1.552	1.684	1.752	1.651	1.696	1.752	0.886
Local	1.487	1.624	1.583	1.598	1.554	1.624	0.886
4D-FED-GNN+	1.393	1.401	1.484	1.422	1.379	***1.484***	***0.678***
FedLink	1.347	1.408	1.395	1.463	1.421	**1.463**	**0.676**

measurements demonstrate FedLink's computational efficiency compared to baseline methods. We also present detailed test AUCs for different experiments, showing AUC score for each country in Table C.2. To ensure model convergence, we monitor AUC scores over increasing training rounds.

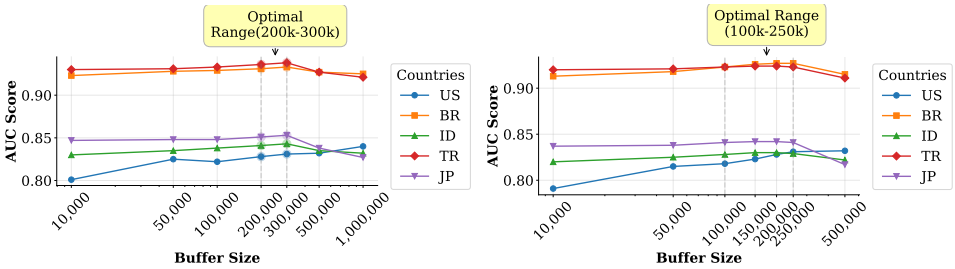

Figure C.3: Buffer Size Effect on AUC scores for five countries(US, BR, ID, TR, JP) on original data (left) and 50% downsampled data (right).

Our experiments use 180 global rounds, as we observe that model performance stabilizes well around this point, as shown in Figure C.4 for the convergence patterns.

D.4 Buffer Selection

As shown in Fig C.3, we performed experiments on varying buffer size and evaluated its effect on model AUC and GPU usage. Detailed buffer size analysis on different countries are as follows:

Table C.2: Test AUCs on different test sets with varied countries combination. Each result is tested over a 30-day period with a 200,000 buffer size. The result is averaged over 10 runs. FedLink consistently achieves the best (bold) or second-best (bold italics) AUC compared with local training and STFL which require significantly more training time.

	US	BR	ID	TR	JP
STFL	***0.842±0.004***	0.927±0.006	0.832±0.006	0.922±0.004	0.828±0.008
Local	0.841±0.008	**0.931±0.007**	**0.836±0.009**	**0.931±0.006**	**0.851±0.008**
4D-FED-GNN+	0.561±0.006	0.595±0.007	0.539±0.012	0.614±0.004	0.588±0.014
FedLink	**0.847±0.003**	***0.930±0.004***	***0.834±0.004***	***0.929±0.003***	***0.842±0.005***
	MX	PH	ES	GB	IT
STFL	***0.876±0.004***	**0.865±0.004**	***0.839±0.003***	**0.824±0.004**	0.836±0.004
Local	0.873±0.006	***0.865±0.009***	0.838±0.009	***0.822±0.010***	***0.837±0.010***
4D-FED-GNN+	0.640±0.002	0.544±0.002	0.546±0.002	0.568±0.002	0.536±0.001
FedLink	**0.877±0.004**	0.865±0.003	**0.839±0.005**	0.821±0.006	**0.837±0.005**
	US	BR	JP	MX	ES
STFL	***0.831±0.003***	***0.929±0.006***	0.832±0.008	0.899±0.005	0.853±0.008
Local	**0.838±0.008**	**0.931±0.007**	**0.843±0.009**	**0.908±0.006**	**0.865±0.008**
4D-FED-GNN+	0.560±0.012	0.594±0.013	0.590±0.012	0.646±0.012	0.547±0.014
FedLink	0.828±0.005	0.929±0.001	***0.835±0.004***	***0.903±0.005***	***0.859±0.003***
	US	IT	GB	TR	ES
STFL	***0.842±0.005***	0.853±0.007	***0.872±0.004***	0.871±0.003	0.854±0.009
Local	**0.843±0.010**	**0.864±0.007**	0.864±0.008	**0.879±0.004**	**0.866±0.007**
4D-FED-GNN+	0.552±0.007	0.563±0.009	0.571±0.011	0.583±0.008	0.547±0.013
FedLink	0.840±0.006	***0.861±0.003***	**0.873±0.007**	***0.874±0.003***	***0.863±0.004***

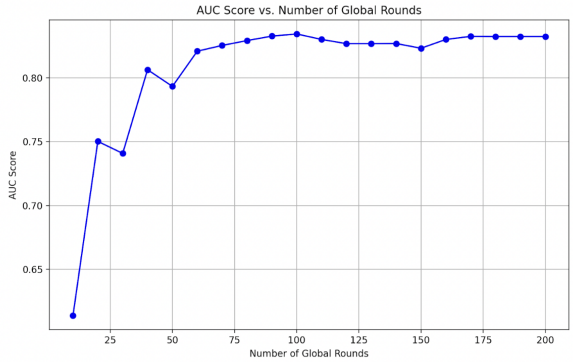

Figure C.4: AUC scores convergence analysis for FedLink over global training rounds

1. For the US dataset, it is the largest country in number of check-ins, and is the only one that will benefit from increasing buffer size even beyond 500K with continuously improved AUC, as it haven't reach its full batch check-in size.
2. For medium-sized countries (BR, TR, MX), the improvement plateaus around 200K, and actual performance decreases beyond 500K.
3. For smaller countries (JP, GB, ES), and the downsampled version of medium sized countries in **Figure C.3(right)**, optimal performance is achieved at even smaller buffer sizes between 100K-150K.

D.5 Ablation Study

We also present detailed experimental results for ablation study for downsampling data in Figure C.5. Table C.3 presents model performance under various data reduction scenarios (50%, 25%, and 2% of original data) across different country combinations.

Table C.3: Test AUCs on down-sampled test sets with varied countries combination with 50%, 25%, 2% size reduction on different country combination's data input. Each result is tested over a 30-day period with a 200,000 buffer size. The result is averaged over 10 runs. FedLink consistently achieves the best (bold) or second-best (bold italics) AUC compared with local training and STFL which require significantly more training time.

	US(50%)	BR(50%)	ID(50%)	TR(50%)	JP(50%)
STFL	***0.836±0.004***	0.915±0.003	0.835±0.006	0.920±0.004	0.804±0.007
Local	0.835±0.008	**0.926±0.008**	**0.8412±0.008**	**0.928±0.005**	**0.843±0.007**
4D-FED-GNN+	0.549±0.007	0.586±0.008	0.529±0.010	0.613±0.003	0.579±0.010
FedLink	**0.843±0.004**	***0.928±0.004***	***0.833±0.003***	***0.917±0.003***	***0.847±0.005***
	US(50%)	BR(50%)	JP(50%)	MX(50%)	ES(50%)
STFL	***0.831±0.004***	***0.929±0.006***	0.832±0.006	0.899±0.005	0.853±0.008
Local	**0.838±0.008**	**0.931±0.007**	**0.843±0.009**	**0.908±0.006**	**0.865±0.008**
4D-FED-GNN+	0.560±0.012	0.594±0.013	0.590±0.012	0.646±0.012	0.547±0.014
FedLink	0.828±0.005	0.929±0.001	***0.835±0.004***	***0.903±0.005***	***0.859±0.003***
	US(25%)	BR(25%)	JP(25%)	MX(25%)	ES(25%)
STFL	0.683±0.006	0.834±0.006	0.720±0.005	0.780±0.008	0.701±0.006
Local	**0.687±0.008**	**0.835±0.007**	**0.735±0.009**	**0.798±0.006**	**0.713±0.008**
4D-FED-GNN+	0.508±0.012	0.519±0.013	0.538±0.012	0.546±0.012	0.509±0.014
FedLink	**0.693±0.005**	**0.837±0.001**	***0.731±0.004***	***0.793±0.005***	***0.709±0.003***
	US(2%)	BR(2%)	JP(2%)	MX(2%)	ES(2%)
STFL	***0.831±0.004***	***0.929±0.006***	0.832±0.006	0.899±0.005	0.853±0.008
Local	**0.838±0.008**	**0.931±0.007**	**0.843±0.009**	**0.908±0.006**	**0.865±0.008**
4D-FED-GNN+	0.560±0.012	0.594±0.013	0.590±0.012	0.646±0.012	0.547±0.014
FedLink	0.828±0.005	0.929±0.001	***0.835±0.004***	***0.903±0.005***	***0.859±0.003***
Method	Full Data	50% Large	50% Small	25% Data	2% Data
STFL	0.913±0.004	0.859±0.005	0.869±0.006	0.744±0.006	0.869±0.006
Local	**0.915±0.008**	**0.870±0.009**	**0.754±0.008**	**0.877±0.008**	**0.877±0.008**
4D-FED-GNN+	0.567±0.009	0.566±0.009	0.587±0.013	0.509±0.014	0.549±0.010
FedLink	***0.916±0.003***	***0.874±0.004***	***0.871±0.004***	***0.802±0.004***	***0.889±0.007***

Table C.4: Test AUCs for only traveled users under the three ablation study methods of FedLink, FedLink-Local and FedLink-NoEmb.

	US	BR	ID	TR	JP
FedLink (FL+Buffer)	**0.808±0.001**	**1.000±0.001**	**0.964±0.036**	**0.833±0.001**	**0.923±0.182**
FedLink-Local	0.552±0.002	0.732±0.158	0.548±0.124	0.531±0.004	0.610±0.215
FedLink-NoEmb	0.759±0.003	0.724±0.086	0.813±0.108	0.783±0.164	0.811±0.007

Table C.4 specifically focuses on traveled users, comparing performance across three variants: FedLink, FedLink-Local, and FedLink-NoEmb. This analysis shows the importance of both federated learning and embedding sharing mechanisms for predicting traveled users. FedLink consistently outperforms other variants on traveled users, demonstrating the effectiveness of cross-client information sharing for users who move between regions.

D.6 TGBL-Wiki

The dataset of TGBL-Wiki includes 157474 edges with timestamps from 0 to 2678373. There are total of 4613 users and 4614 items, with 133892 user-item interactions.

To evaluate temporal performance across different time periods, we partition the TGBL-Wiki dataset into two temporal subsets based on timestamp ordering: an **earlier period** containing the first half of interactions (timestamps 0 to 1339186) and a **later period** containing the second half (timestamps 1339187 to 2678373). This temporal split allows us to assess how federated learning methods perform on older versus newer interaction patterns, providing insights into temporal generalization capabilities.

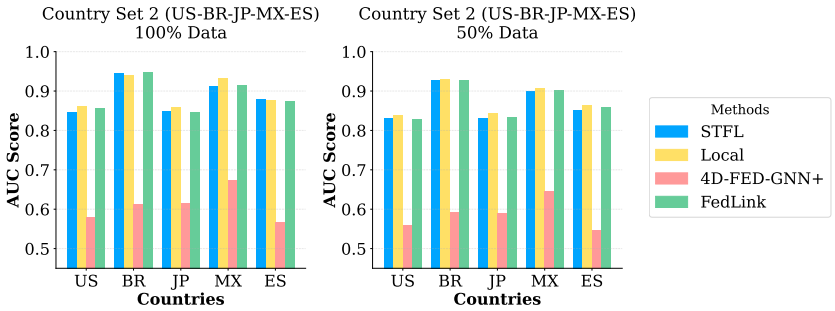

Figure C.5: Performance comparison under data downsampling (100% vs 50%). Results demonstrate that FedLink maintains consistent AUC scores even with down-sampled data across both country sets. FedLink maintains comparable AUC scores with 50% data size across both country sets, while achieving reduced GPU memory consumption and faster training time.

E Example Use Cases

E.1 Link Prediction on Tabular Data

Most user data is stored in tabular format, where each row in the table represents information on a user information (e.g., user IP address, location, website, and purchased product). User information is also updated with time. However, entries in this table may be missing, e.g., due to faulty data collection. By modeling the user table information as a dynamic graph, we can perform link prediction on the tabular data and fill in the missing part of the table, allowing us to more easily use it for various tasks (e.g., purchase behavior prediction).

E.2 Website Behavior Prediction

Predicting websites to be visited by users allows dynamic pre-caching of the site content, reducing communication costs and response latency. By modeling visits as a user-website bipartite graph, we can predict frequently visited websites. Such visit behavior is cyclic and dynamic, requiring regular updates and on-time predictions.

E.3 User Identity Verification

Identity verification helps to validate users' product subscriptions. By modeling users and their behavior as nodes in a dynamic graph, we can detect anomalous behavior and classify nodes as "good" users and "bad" actors.