Membership Inference Attacks on Graph Neural Networks Using Synthetic Data: A Privacy Vulnerability Analysis

Prateek Rathod
Department of Computer Science
Master's Program in Computer Science

September 12, 2025

Abstract

Membership Inference Attacks (MIAs) create privacy risks for machine learning models by finding out if specific data points were used during training. This project studies an attack against Graph Neural Networks (GNNs) that uses fake graph data. This removes the attacker's need to access real training data. We change existing MIA methods to work with fake graphs made by the DLGrapher project [**?**]. We test attack success on four GNN types: Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), GraphSAGE, and Simple Graph Convolution (SGC).

We use real-world data from Twitch and Event platforms. Our tests show that fake data can work for membership inference attacks but performs worse than using real data for shadow model training. We create the TSTS (Train on Subgraph, Test on Subgraph) method to handle computer limits while keeping attacks working. Results show big differences in how vulnerable each model is. SGC is most vulnerable, GraphSAGE and GAT are moderately vulnerable, while GCN is most resistant with nearly random results.

Our findings show important privacy problems in GNN use and stress the need for privacy protection beyond just limiting data access. This work helps understand the trade-off between privacy and usefulness in graph learning systems and gives ideas for building strong defenses against membership inference attacks.

Contents

1 Introduction

Membership Inference Attacks (MIAs) are a serious privacy problem in machine learning systems. They let attackers find out if specific data points were used during model training [?]. This creates big risks in areas like healthcare, finance, and social networks, where training data often has personal information. Traditional MIA methods need access to real training data. This project looks at a new type of attack that uses fake graph data instead.

Graph Neural Networks (GNNs) are powerful tools for learning from graph-structured data. They are used in social network analysis, recommendation systems, and biological networks [?, ?, ?]. However, we don't know much about how vulnerable they are to membership inference attacks [?], especially when attackers can't access the original data. This project fills this gap by creating and testing MIA methods that use fake graph generation. We use DLGrapher [?] to create fake graph structures that look like real-world data.

The main contributions of this work are: (1) changing existing MIA methods to work with fake graph data, (2) testing multiple GNN types including Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), GraphSAGE, and Simple Graph Convolution (SGC), (3) creating better feature engineering methods for improved attack performance, and (4) testing using real-world data from Twitch and Event platforms. Our results show that fake data can replace real training samples in shadow model building. This achieves attack success rates similar to traditional methods while needing much less knowledge from the attacker.

2 Background

2.1 Membership Inference Attacks

Membership Inference Attacks use the fact that machine learning models often overfit their training data. This creates different patterns between members (training samples) and non-members (unseen samples) [?, ?]. The attack has three main parts:

Target Model: The victim GNN model trained on private graph data that the attacker wants to attack. This model creates probability outputs that accidentally leak membership information through small statistical patterns.

Shadow Models: Models that are similar to the target model, trained on data that should follow similar patterns. These models copy the target's behavior. This lets the attacker see membership patterns without direct access to the target's training process.

Attack Model: A binary classifier trained to tell the difference between members and non-members based on model outputs. It learns decision rules from shadow model behaviors and uses them on target model outputs.

2.2 Graph Neural Networks

GNNs extend deep learning to graph data by combining information from node neighborhoods through message-passing. The models we test in this work include:

GCN: Uses graph convolutions with localized filters to combine features via $H^{(l+1)} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)})$ [?].

GAT: Uses attention mechanisms to learn how much weight to give each neighbor, computing $\alpha_{ij} = \text{softmax}_j(\text{LeakyReLU}(a^T[Wh_i||Wh_j]))$ [?].

GraphSAGE: Samples fixed-size neighborhoods and combines features through learnable functions (mean, LSTM, or pooling) [?].

SGC: Simplifies GCN by removing nonlinearities between layers, reducing to $Y = \text{softmax}(\tilde{A}^K XW)$ [?].

3 Methodology

This project introduces a new approach to membership inference attacks by using fake graph data made through DLGrapher. This removes the attacker's need for real training data access. Our method is different from traditional MIA approaches through its unique data handling and attack process.

3.1 Attack Architecture

Our attack framework keeps the three-model structure while adding fake data generation as a key part. The target model works on real graph data, while shadow models only use fake graphs that copy the original data patterns. This difference tests whether similar data patterns are enough for successful membership inference.

3.2 Data Pipeline

We handle computer limits of fake graph generation through a subgraph-based approach:

Data Partitioning: Each dataset has three parts: (1) training subgraphs for target model training, (2) non-training subgraphs from the same data pattern for testing, and (3) fake subgraphs made via DLGrapher. Each part has 256 subgraphs with 120 nodes, making sure there is minimal overlap between subgraphs.

Bridge Module: A custom data loader handles binary pickle files containing (DataFrame, NetworkX Graph) pairs, converting them to PyTorch Geometric Data objects that work with GNN implementations.

Adapter Module: Implements the TSTS (Train on Subgraph, Test on Subgraph) method, managing data splits and making sure target and shadow model datasets are properly separated.

3.3 Attack Execution Pipeline

The attack works through seven steps:

1. **Data Preparation:** Load and prepare graph data from pickle files
2. **Target Model Training:** Train GNN on real training subgraphs
3. **Shadow Model Training:** Train same architecture on fake subgraphs
4. **Feature Engineering:** Extract and improve graph features including degree centrality, clustering coefficients, and spectral properties
5. **Attack Model Training:** Train binary classifier on shadow model outputs with known membership labels
6. **Attack Execution:** Apply trained attack model to target model outputs
7. **Performance Evaluation:** Calculate accuracy, AUROC, precision, recall, and F1 scores

4 Implementation Details

4.1 Datasets

We use two real-world social network datasets, each with unique features and challenges for membership inference:

Twitch Dataset: Comes from the Twitch streaming platform, containing user interaction graphs with features including `views`, `mature`, `life_time`, `created_at`, `updated_at`, `dead_account`, `language`, and `affiliate` (target variable). The dataset captures social dynamics and content consumption patterns.

Event Dataset: Comes from event-based social platforms, featuring user attributes such as `locale`, `birthyear`, `gender` (target variable), `joinedAt`, and `timezone`. This dataset represents demographic-based social connections.

Each dataset is prepared into 256 subgraphs containing 120 nodes with minimal overlap between subgraphs, stored as pickle files containing (pandas.DataFrame, networkx.Graph) pairs. The binary classification task matches the original MIA framework requirements.

4.2 Data Integration Pipeline

The `bridge.py` module handles data format conversion, changing pickle-stored subgraphs into PyTorch Geometric Data objects. This conversion keeps graph structure while making sure it works with GNN implementations. The module uses lazy loading to optimize memory usage for large experiments.

4.3 Feature Engineering

The `rebmi_adapter.py` module implements better feature extraction beyond raw node attributes:

- **Structural Features:** Degree centrality, clustering coefficient, betweenness centrality
- **Spectral Features:** Eigenvalue-based graph properties, Laplacian eigenmaps
- **Neighborhood Features:** Aggregated statistics from k-hop neighborhoods
- **Temporal Features:** For time-stamped attributes in Twitch dataset

Feature standardization uses z-score normalization with clipping to handle outliers, important for different types of graph data.

4.4 Attack Model Enhancement

The improved attack model architecture includes several improvements:

- **Multi-layer Perceptron:** Three hidden layers with [128, 64, 32] neurons
- **Regularization:** Dropout (p=0.3) and L2 weight decay ($\lambda = 0.001$)
- **Ensemble Methods:** Combines predictions from multiple shadow models
- **Confidence Calibration:** Temperature scaling for posterior probability adjustment

5 Results and Experimentation

We ran many experiments to test attack performance across different GNN architectures, datasets, and settings. Each experiment runs multiple times with different random seeds to ensure statistical significance.

5.1 Experimental Setup

Hardware: NVIDIA GPU with CUDA support, 32GB system memory
Software: PyTorch 1.13, PyTorch Geometric 2.2, Python 3.11
Training: 300 epochs for target/shadow models, early stopping with patience=20
Metrics: Attack accuracy, AUROC, precision, recall, F1-score

5.2 Attack Performance

Results show different vulnerability levels across GNN architectures:

GCN Performance: Showed strong resistance with 49.5% ± 0.4% attack accuracy on Twitch dataset and 49.7% ± 0.3% on Event dataset. The near-random performance shows effective protection against membership leakage.

GAT Performance: Showed moderate vulnerability with 49.1% ± 0.8% accuracy on Twitch and 51.7% ± 3.8% on Event. High variance on Event dataset (45.4%-54.6%) suggests dataset-dependent attack patterns.

GraphSAGE Performance: Showed consistent moderate vulnerability with 50.1% ± 0.5% accuracy on Twitch and 50.9% ± 0.4% on Event, with neighborhood sampling creating patterns that can be exploited.

SGC Performance: Most vulnerable architecture with 52.2% ± 0.4% accuracy on Twitch and very high 64.9% ± 1.8% on Event dataset, showing significant privacy risks due to its simple architecture.

5.3 Synthetic Data Effectiveness

Comparing fake and real shadow training data shows important insights:

Baseline Comparisons: Traditional datasets (Cora: 70.5%, CiteSeer: 83.4%) show much higher vulnerability than our custom datasets, validating our experimental setup.

Synthetic vs Real Shadow Data:

- Event dataset: 61.4% (real) vs 49.7% (synthetic) for GCN
- Twitch dataset: 48.5% (real) vs 49.5% (synthetic) for GCN
- GAT Event: 60.7% (real) vs 51.7% (synthetic)
- GAT Twitch: 49.4% (real) vs 49.1% (synthetic)

Key Finding: Fake data reduces attack effectiveness compared to real shadow training data, achieving about 60-70% of baseline performance while still allowing viable attacks.

6 Evaluation and Analysis

6.1 Statistical Analysis

Results show clear vulnerability patterns that depend on architecture:

Vulnerability Ranking:

1. **SGC (Highest Risk):** 52.2%-64.9% attack accuracy

2. **GraphSAGE (Moderate):** 50.1%-50.9% attack accuracy
3. **GAT (Variable):** 49.1%-51.7% with high variance
4. **GCN (Most Resistant):** 49.5%-49.7% near-random performance

Dataset Impact: Event dataset shows consistently higher vulnerability across all architectures, with SGC achieving 64.9% attack accuracy compared to 52.2% on Twitch.

Member vs Non-Member Classification: Extreme patterns observed with some runs achieving 97%+ accuracy on members but <5% on non-members, indicating model overfitting creates distinguishable patterns.

6.2 Performance Variance Analysis

Consistency Rankings:

- **Most Consistent:** GCN ($\sigma < 0.4\%$) and SGC on Twitch
- **High Variance:** GAT on Event dataset ($\sigma = 3.8\%$)
- **Architecture Dependency:** Attention mechanisms (GAT) show dataset-sensitive performance

7 Discussion

The results show that fake graph data can effectively enable membership inference attacks without direct access to real training data patterns. This finding has important implications for privacy protection in graph learning systems.

7.1 Privacy Implications

The experimental results show important insights for GNN deployment in privacy-sensitive contexts:

Architecture Selection Impact: The choice of GNN architecture greatly affects privacy risk, with SGC showing 3x higher vulnerability than GCN on certain datasets.

Synthetic Data Threat: While fake shadow training data reduces attack effectiveness, it still enables viable membership inference attacks without requiring access to real training data patterns.

Dataset Characteristics: Demographic features (Event dataset) appear more exploitable than behavioral features (Twitch dataset), suggesting privacy risks vary by application area.

7.2 Defense Strategies

Based on the experimental findings, several defense strategies emerge:

Architecture-Based Defenses:

- **Prefer GCN:** Choose GCN over SGC for privacy-sensitive applications (49.7% vs 64.9% attack success)
- **Avoid SGC:** SGC's simplified architecture significantly increases vulnerability
- **Consider GAT Carefully:** High variance suggests unpredictable privacy behavior

Additional Protections:

- **Enhanced Regularization:** GCN's inherent regularization provides natural privacy protection
- **Output Perturbation:** Adding calibrated noise to posteriors can obscure membership signals [**?**]
- **Dataset Considerations:** Demographic features require stronger privacy protections than behavioral data

7.3 Limitations

Several constraints limit the scope and generalizability of findings:

Experimental Limitations:

- **Dataset Scope:** Limited to two social network datasets; broader domain evaluation needed
- **Synthetic Data Quality:** DLGrapher quality not directly measured or compared to alternatives
- **Subgraph Approach:** Fixed 120-node subgraphs may not represent full-graph scenarios
- **Architecture Parameters:** Fixed hyperparameters may not be optimal across all models

Methodological Constraints:

- **Attack Sophistication:** Basic attack model; advanced techniques might show different results
- **Defense Evaluation:** No comparison with state-of-the-art privacy-preserving techniques
- **Scalability:** Computational cost limits extensive hyperparameter exploration

8 Conclusion

This project successfully shows membership inference attacks on Graph Neural Networks using fake data, revealing moderate attack success rates (49%-65% across architectures and datasets) while removing the attacker's need for real training data access. Changing existing MIA frameworks to work with DLGrapher-generated fake graphs opens new attack methods that organizations must consider when using GNN systems.

Key contributions include creating the TSTS methodology for subgraph-based attacks, complete evaluation across four GNN architectures (GCN, GAT, GraphSAGE, SGC), and testing on real-world social network datasets. The experimental results show a clear vulnerability ranking: SGC shows highest risk (52.2%-64.9% attack accuracy), followed by GraphSAGE (50.1%-50.9%) and GAT (49.1%-51.7%), while GCN shows strong resistance with near-random performance (49.5%-49.7%).

The findings show important privacy implications for GNN deployment. While fake data-based attacks are less effective than traditional approaches, they show that attackers can infer membership without access to real training data. The results stress the critical importance of architecture selection in privacy-sensitive applications, with GCN providing much better privacy protection than SGC (49.7% vs 64.9% attack success on Event dataset). As graph learning continues to expand into critical areas involving personal data, these architectural considerations become very important for maintaining user privacy and regulatory compliance.

9 Future Work

Several promising directions extend this research:

Advanced Synthetic Generation: Looking into other graph generation methods beyond DLGrapher, including GANs and VAEs specifically designed for graph structures.

Adaptive Attacks: Creating attacks that dynamically adjust to target model defenses through reinforcement learning or adversarial training.

Cross-Domain Evaluation: Extending evaluation to biological networks, knowledge graphs, and recommendation systems to test generalizability.

Defense Development: Designing GNN-specific defense mechanisms that maintain usefulness while provably limiting membership leakage.

Theoretical Analysis: Creating formal privacy guarantees and finding theoretical limits on membership inference success rates.

Federated Learning: Looking into membership inference in federated GNN settings where data remains distributed across multiple parties.