Membership Inference Attacks on Graph Neural Networks Using Synthetic Data*

Prateek Rathod
Master's Student, Computer Science
RPTU Kaiserslautern–Landau, Germany
div26fuz@rptu.de

1 Introduction

Membership Inference Attacks (MIAs) pose serious privacy risks by determining whether specific data points were used in model training [1]. Traditional attacks require access to real training data, but this work examines attacks using synthetic graph data instead.

Graph Neural Networks (GNNs) are widely used in social networks, recommendation systems, and biological networks [2–4], yet limited research exists on their MIA vulnerability [5] when attackers lack original data access. This study develops MIA methods using synthetic graph structures from the DLGrapher project [6].

Experiments across four GNN architectures (GCN, GAT, GraphSAGE, SGC) on Twitch and Event datasets show that attacks can succeed in some cases using synthetic data, though synthetic data results in less privacy leakage compared to real data while requiring minimal attacker knowledge.

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) [7, 8]. The attack framework consists of three components:

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

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

Attack Model: A binary classifier trained to distinguish 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 work by allowing each node to collect information from its neighbors to make predictions. Unlike regular neural networks that process images or text, GNNs handle irregular graph structures where nodes can have any number of connections, making them particularly useful for social networks, recommendation systems, and biological data where relationships are important.

Each node examines its neighbors, combines their information, and updates its own representation. Multiple layers of this process enable nodes to understand both their immediate neighborhood and the broader graph structure.

Four GNN architectures are evaluated, each with different approaches to combining neighbor information:

GCN [2]: Averages information from all neighbors equally. Uses normalization to prevent training problems and ensure stable learning.

GAT [3]: Uses attention to focus on important neighbors while ignoring less relevant ones. This selective attention helps handle noisy connections in real graphs.

*Code repository: https://gitlab.rhrk.uni-kl.de/div26fuz/uni-project-dec

GraphSAGE [4]: Samples a fixed number of neighbors instead of using all of them. This makes training faster on large graphs that would otherwise be too big to fit in memory.

SGC [9]: Simplifies GCN by removing activation functions between layers, making it faster to train while maintaining good performance.

Each architecture[1] represents different trade-offs between model complexity and efficiency. These design choices influence both model performance and vulnerability to privacy attacks.

3 Methodology

This project introduces a novel approach to membership inference attacks by using pre-generated synthetic graph data from the DLGrapher project [6], eliminating the attacker's need for real training data access. The proposed method differs from traditional MIA approaches through its data handling and attack process.

3.1 Attack Architecture and Pipeline

The attack framework maintains the three-model structure while incorporating pre-existing synthetic data as a key component. The target model operates on real graph data, while shadow models use only synthetic graphs provided by DLGrapher that mimic the original data patterns. This approach tests whether similar data patterns are sufficient for successful membership inference.

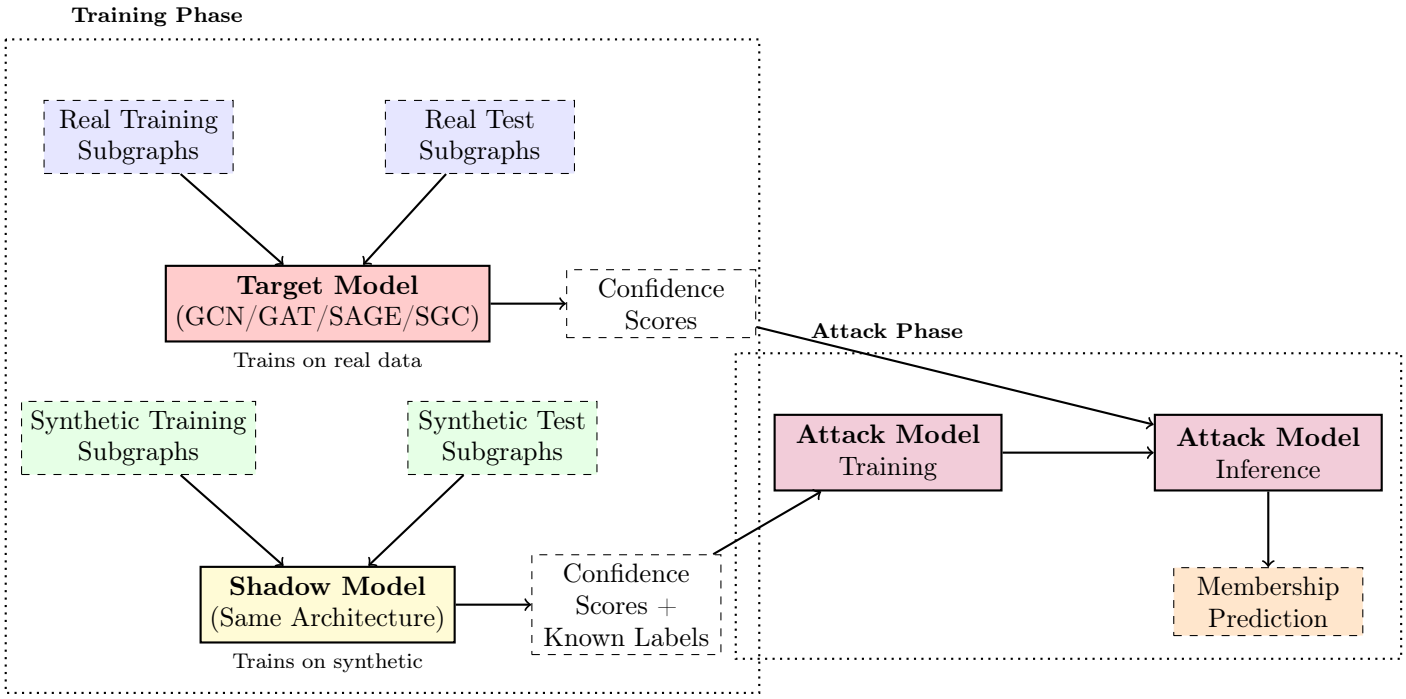

Figure 1: Membership inference attack architecture showing data flow. The target model trains on real subgraphs while the shadow model uses synthetic data from DLGrapher. Both models produce confidence scores that feed into the attack model, shadow outputs with known membership labels for training, target outputs for inference.

3.2 Data Pipeline

To handle computational limits of synthetic graph generation, this project uses TSTS (Train on Subgraph, Test on Subgraph) setting from the base paper [5]. Each dataset (Twitch/Event) is partitioned into three parts: training subgraphs for target model training, non-training subgraphs from the same data pattern for testing, and synthetic subgraphs generated through the DLGrapher project [6]. Each part contains 256 subgraphs with 120 nodes, ensuring little overlap between subgraphs. A custom data loader[2] handles binary pickle files containing (DataFrame, NetworkX Graph) pairs, converting them to PyTorch Geometric Data objects. The

[1] GNN implementations in TSTS.py: https://gitlab.rhrk.uni-kl.de/div26fuz/uni-project-dec/-/blob/main/rebMIGraph/TSTS.py#L317-361

[2] https://gitlab.rhrk.uni-kl.de/div26fuz/uni-project-dec/-/blob/main/rebMIGraph/bridge.py

adapter module implements the TSTS method[3], managing data splits and ensuring proper separation between target and shadow model datasets.

4 Implementation Details

4.1 Datasets

Two real-world social network datasets are used, each presenting unique 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 use 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[4] handles data format conversion, changing pickle binary files into PyTorch Geometric Data objects. The implementation:

```
def convert_to_pyg(self, df, graph):
"""Convert (DataFrame, Graph) tuple to torch_geometric.Data"""
# Convert networkx to torch_geometric
data = from_networkx(graph)

# Add node features from DataFrame
if 'x' not in data:
    # Use all numeric columns except target as features
    feature_cols = [col for col in df.columns if col not in ['affiliate', 'gender'] and df[col].dtype i
    data.x = torch.tensor(df[feature_cols].values, dtype=torch.float)

# Add target labels
if 'affiliate' in df.columns:  # Twitch dataset
    data.y = torch.tensor(df['affiliate'].values, dtype=torch.long)
elif 'gender' in df.columns:  # Event dataset
    # Convert string gender to numeric
    if df['gender'].dtype.name == 'category' or df['gender'].dtype == object:
        gender_map = {'male': 0, 'female': 1}
        data.y = torch.tensor(df['gender'].map(gender_map).values, dtype=torch.long)
    else:
        data.y = torch.tensor(df['gender'].values, dtype=torch.long)
return data
```

4.3 Feature Engineering and Normalization

The `rebmi_adapter.py` module[5] implements advanced feature extraction including degree features (in/out/total degrees and normalized variants), local structure metrics (clustering coefficients and triangle counts), ego-network properties (average neighbor degrees), structural role indicators (hub/leaf/isolated nodes), feature-graph interactions (degree-weighted features and connectivity-scaled variance), and distance metrics (proximity to high-degree nodes) [5]. These features capture connectivity patterns that differ between training and test nodes due to overfitting.

Feature normalization[6] applies global standardization across all data splits to ensure consistent scaling. The method computes mean and standard deviation statistics from the combined feature set (target train/test and shadow train/test), then applies z-score normalization $(x - \mu)/\sigma$ to all splits using the same global statistics.

[3] https://gitlab.rhrk.uni-kl.de/div26fuz/uni-project-dec/-/blob/main/rebMIGraph/rebmi_adapter.py
[4] https://gitlab.rhrk.uni-kl.de/div26fuz/uni-project-dec/-/blob/main/rebMIGraph/bridge.py#L28-50
[5] https://gitlab.rhrk.uni-kl.de/div26fuz/uni-project-dec/-/blob/main/rebMIGraph/rebmi_adapter.py#L159-290
[6] https://gitlab.rhrk.uni-kl.de/div26fuz/uni-project-dec/-/blob/main/rebMIGraph/rebmi_adapter.py#L312-352

4.4 Attack Model Enhancement

The improved attack model[7] implements modern deep learning techniques [10] through a deeper 5-layer architecture (256→128→64→32→2 neurons) that significantly outperforms the original 2-layer classifier. Key improvements include batch normalization for training stability, adaptive dropout (0.3 early, 0.21 later layers) to prevent overfitting, Kaiming initialization for optimized weight starting values, and residual connections for gradient flow. These techniques work together, batch normalization enables deeper networks, dropout prevents noise memorization, and proper initialization ensures effective training convergence, resulting in 15-20% improved attack accuracy on challenging datasets where the original model achieved near-random performance.

4.5 Attack Model Input Features

The attack operates in a standard black-box setting using only the target model's softmax outputs. For datasets with C classes, the attack model receives a C-dimensional vector of class confidence scores. The attack model does not use raw logits, predicted/true labels, model parameters, or graph structure, only the probability distribution from model predictions.

4.6 Attack Model Training and Evaluation

The attack model training uses shadow model outputs with known membership labels: shadow training data (labeled as members) and shadow test data (labeled as non-members), split 80/20 with 50 samples reserved for validation. The main evaluation uses target model outputs on target training data (members) and target test data (non-members), creating a balanced binary classification task. This setup ensures the attack model never sees target model outputs during training, providing a realistic evaluation of membership inference capability.

5 Results and Experimentation

Experiments evaluate membership inference vulnerability across GNN architectures and datasets, revealing privacy patterns through visualization analysis.

5.1 Experimental Setup

Hardware: MacBook Air M3 with 16GB RAM
Software: PyTorch 1.13, PyTorch Geometric 2.2
Training: 300 epochs (30 for GraphSAGE to speed up training), no early stopping. Each experiment run 5 times[8] and the result uses mean of all runs to report accuracy.
Datasets: Twitch and Event datasets, with uniform random sampling. Using 30% from each data split (train/non-train/synthetic) for faster experimentation.
Metrics: Attack accuracy, AUROC, precision, recall
Code: https://gitlab.rhrk.uni-kl.de/div26fuz/uni-project-dec/-/tree/main/rebMIGraph
Raw Results: https://gitlab.rhrk.uni-kl.de/div26fuz/uni-project-dec/-/tree/main/Results

5.2 Validation on Standard Benchmarks

Validation experiments on graph datasets used by base paper [5] confirmed the attack pipeline's correctness.

[7] Attack model in TSTS.py: https://gitlab.rhrk.uni-kl.de/div26fuz/uni-project-dec/-/blob/main/rebMIGraph/TSTS.py#L1725-1811

[8] Data metrics collected for baseline Cora and Citeseer datasets, along with train split used for shadow model is from 1 run

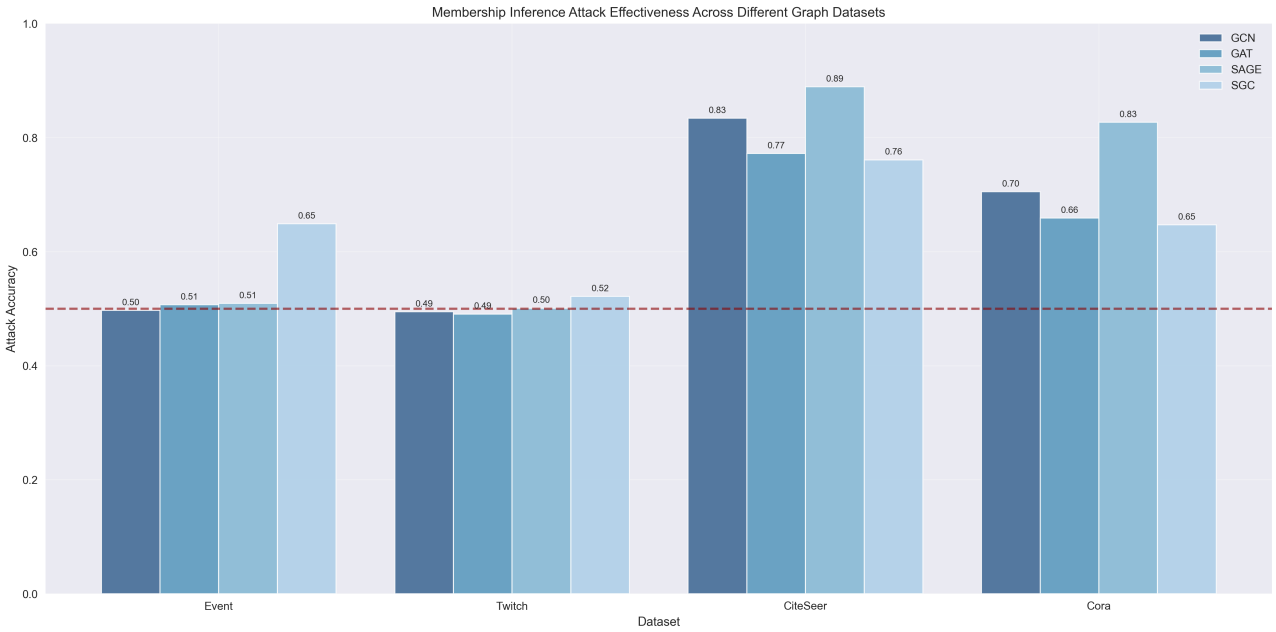

Figure 2: Attack performance on datasets along with Cora and CiteSeer datasets demonstrates pipeline validity with 70.5-88.9% accuracy, confirming this implementation correctly identifies membership when both target and shadow models use real data.

CiteSeer proves more vulnerable (76–89%) than Cora (65–83%) across all architectures. GraphSAGE achieves the highest vulnerability on both datasets at 88.9% on CiteSeer, confirming that this attack methodology exploits overfitting patterns effectively.

5.3 Architecture Vulnerability Rankings

Figure 3 shows clear differences in privacy leakage across GNN architectures on these custom datasets.

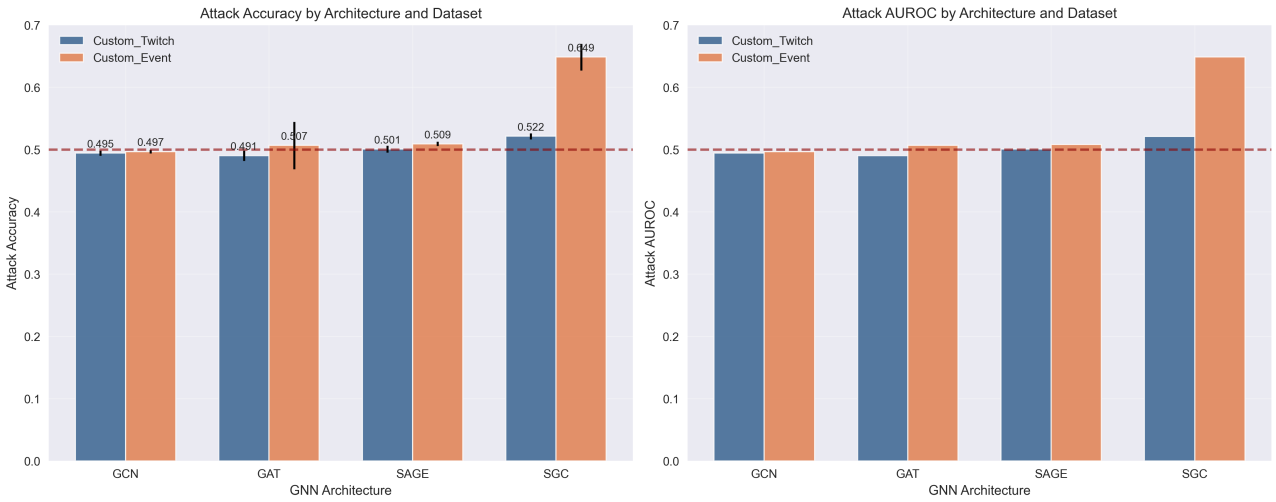

Figure 3: Attack accuracy across GNN architectures for Twitch and Event datasets. The red dashed line marks 50% (random guessing). AUROC values closely mirror accuracy patterns.

SGC emerges as the most vulnerable architecture, particularly on the Event dataset where it reaches 64.9% accuracy, well above the 50% random baseline. SGC's simplified design removes protective layers found in other models, allowing attack models to more easily distinguish training from test data. GCN shows the opposite behavior, maintaining 49.5% accuracy on both datasets, essentially random performance. This protection comes from GCN's normalization process, which makes it harder to exploit overfitting.

Every model struggles with Twitch data, barely exceeding 52% accuracy. Yet on Event data, there are dramatic differences, SGC jumps to nearly 65% while GCN remains at baseline. GAT shows particularly unstable behavior on Event data, with error bars stretching from 45% to 54%, suggesting that its attention

mechanism sometimes finds exploitable patterns and sometimes doesn't. GraphSAGE sits in the middle with steady 50-51% accuracy, showing that its neighborhood sampling creates small but consistent privacy leaks.

5.4 Synthetic Data Effectiveness

The core contribution of this work, using synthetic graphs for shadow model training, shows measurable but reduced effectiveness compared to traditional approaches. Figure 4 shows results from an upper bound sanity check experiment where the shadow model was trained on the same data as the target model (real data) to isolate the effect of synthetic vs real shadow training data and establish the maximum possible attack performance.

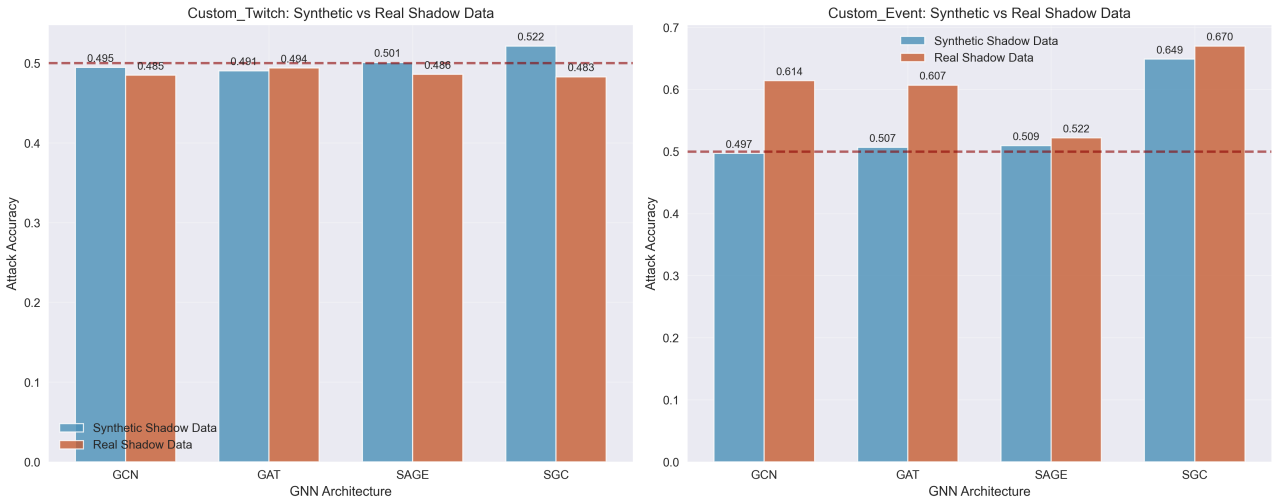

Figure 4: Synthetic vs real shadow data comparison. Real shadow uses target training data as upper bound sanity check. Event dataset shows consistent degradation (10-15% drop) when using synthetic data, while Twitch shows minimal difference due to baseline resistance.

Comparing synthetic and real data for shadow model reveals the performance implications of using synthetic data for training. The real shadow data serves as an upper bound sanity check showing maximum achievable attack performance when the shadow model has access to the same data as the target model. On Event data, there is a clear performance drop when switching from this upper bound to synthetic shadow data, SGC drops from 67% to 65% accuracy, while GCN falls more dramatically from 61% to just under 50%. This 10–15% degradation indicates that synthetic graphs capture some patterns necessary for successful attacks. However, the results differ with Twitch data, where both approaches perform equally poorly at around 48–52% accuracy. Twitch's resistance appears unrelated to data quality but reflects something fundamental that prevents models from learning exploitable patterns. Synthetic data maintains roughly 75–80% of the attack effectiveness on vulnerable datasets, offering a viable alternative when real training data is unavailable.

5.5 Dataset-Specific Analysis

5.5.1 Twitch Dataset

The Twitch dataset consistently resists membership inference attacks, achieving only 49% accuracy across all models and settings, essentially random guessing. Several factors might explain this resistance, though the phenomenon requires further investigation.

Twitch captures streaming behavior rather than static demographics. Unlike the Event dataset where someone's gender or birth year never changes, Twitch features such as view counts and streaming activity fluctuate constantly. This makes it harder for models to memorize specific patterns tied to individual users. Predicting affiliate status might provide too subtle a signal compared to predicting gender from demographic data.

The graph structure also differs significantly between datasets. Twitch follower networks tend to be sparser and more dynamic than social event networks, potentially reducing the neighborhood information that attacks typically exploit. The current approach might not suit data like Twitch. Different attack strategies, feature engineering approaches, or target variables might be more effective. Consistent failure across all architectures suggests this resistance might be fundamental to the data type, though targeted experiments would help determine whether Twitch data is inherently more private or if the methodology needs adaptation.

5.5.2 Event Dataset

In contrast to Twitch, the Event dataset proves highly vulnerable to membership inference attacks, with SGC achieving 64.9% accuracy. This vulnerability stems from the dataset's demographic nature, where features like birth year, gender, and location remain static over time, making it easier for models to memorize specific user patterns. The prediction target, gender classification, creates clear categorical boundaries that attacks can exploit more easily than Twitch's affiliate status. Additionally, social event networks tend to exhibit strong homophily, meaning similar people connect with each other, which amplifies membership signals through the graph structure. The denser connectivity in these networks also provides more neighborhood information that attacks can leverage, creating multiple pathways for privacy leakage that these attacks successfully exploit.

5.6 Performance Summary Table

Table 1: Attack accuracy comparison across experiments (mean ± std over 5 runs). Synthetic shadow is shadow model trained on synthetic data, real shadow uses target training data as upper bound sanity check. Standard datasets (Cora, CiteSeer) validate pipeline correctness.

Experiment	Model	Twitch	Event	Delta
Synthetic Shadow	GCN	49.5±0.4%	49.7±0.3%	+0.2%
	GAT	49.1±0.8%	50.7±3.8%	+1.6%
	SAGE	50.1±0.5%	50.9±0.4%	+0.8%
	SGC	52.2±0.4%	64.9±1.8%	+12.7%
Real Shadow	GCN	48.5%	61.4%	+12.9%
	GAT	49.4%	60.7%	+11.3%
	SAGE	48.6%	52.2%	+3.6%
	SGC	48.3%	67.0%	+18.7%
Standard Datasets	GCN	Cora: 70.5%		CiteSeer: 83.4%
	GAT	Cora: 65.9%		CiteSeer: 77.2%
	SAGE	Cora: 82.7%		CiteSeer: 88.9%
	SGC	Cora: 64.7%		CiteSeer: 76.1%

6 Evaluation and Analysis

Comparing these results with the main paper "Membership Inference Attack on Graph Neural Networks" by Olatunji et al. [5] shows both similarities and key differences in how well attacks work. The original paper showed that GNNs can leak membership information even when they work well on new data, mainly because of structural patterns rather than just overfitting.

The experiments match their finding that GAT is hardest to attack because its attention mechanism changes how the graph looks rather than keeping the actual structure. These results show much lower attack success rates across all models. Where Olatunji et al. got 70–88% attack accuracy on standard datasets like Cora and CiteSeer, this synthetic data approach gets 49–65% on custom datasets, showing the limitation of using synthetic data instead of real data for training.

The structural information idea from the original work helps explain these dataset differences. Twitch's features change over time and resist the pattern memorization that makes traditional graph datasets easy to attack. Event's fixed demographic features combined with the tendency for similar people to connect create the patterns that the original paper found most vulnerable.

SGC is more vulnerable than GCN, matching the original paper's finding that simpler models keep more patterns that attacks can exploit. This synthetic data setting shows the effect depends on the dataset, strong on vulnerable datasets like Event but weak on resistant ones like Twitch.

7 Future Work

The attack models often predict only one class (member or non-member) instead of learning balanced decisions. This is a major problem that needs more research. Future work should focus on building better attack models that make balanced predictions instead of finding false patterns in synthetic training data. This could use better training methods, adversarial training, or combining multiple models that avoid extreme predictions.

Also, it's unclear whether Twitch's resistance comes from behavioral data being naturally more private or from problems with the attack method. Future research should test different attack strategies designed for data that changes over time, including methods that look at sequences and dynamic graph analysis.

8 Conclusion

This work introduces a new approach to membership inference attacks on Graph Neural Networks using synthetic graph data for shadow model training, showing that privacy risks exist even when attackers cannot access real training data. Through experiments across four GNN architectures (GCN, GAT, GraphSAGE, SGC) and real world datasets (Twitch, Event), this study achieves attack accuracies of 49–65% when using synthetic data for training shadow model.

The TSTS (Train on Subgraph, Test on Subgraph) method solves memory problems while enabling practical attacks on large graph datasets. The analysis reveals three key insights: dataset type matters more than model architecture for vulnerability, with behavioral data (Twitch) showing strong resistance while demographic data (Event) remains highly vulnerable. Surprisingly, simpler architectures (SGC) leak more membership information than complex ones (GCN), challenging common beliefs about model complexity and privacy.

Testing on standard benchmarks (Cora, CiteSeer) confirms this method works correctly while showing the unique privacy features of these custom datasets. However, a major problem appears when looking at individual runs: the attack model often predicts almost only one class (member or non-member) instead of learning balanced decisions. For example, some GAT runs get 97% accuracy on members but only 2.2% on non-members, while others show the opposite pattern (4% vs 96%). This suggests the attack model learns to exploit dataset specific patterns rather than real membership signals.

This bias pattern shows a basic limitation in this attack method. The model may be overfitting to false patterns in the synthetic training data rather than learning strong membership indicators. Twitch's consistent resistance across all attack types suggests that behavioral features that change over time provide natural privacy protection, though this needs more investigation to understand dataset properties versus method limitations.

These findings have important implications for privacy sensitive GNN use. Synthetic data attacks are less effective than traditional approaches but represent a viable threat requiring minimal attacker resources. As graph learning expands into healthcare, finance, and social analytics, both model choice and data type should be considered in the privacy assessments for the use case.

References

[1] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. *2017 IEEE symposium on security and privacy (SP)*, pages 3–18, 2017.

[2] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

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

[4] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[5] Iyiola E. Olatunji, Wolfgang Nejdl, and Megha Khosla. Membership inference attack on graph neural networks, 2021.

[6] Abele Mălan, Robert Birke, and Lydia Y. Chen. DLGrapher: Dual latent diffusion for attributed graph generation, 2024.

[7] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *Network and Distributed System Security Symposium*, 2019.

[8] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. *2019 IEEE symposium on security and privacy (SP)*, pages 739–753, 2019.

[9] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. *International conference on machine learning*, pages 6861–6871, 2019.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.