Exercise 2 Report: Stealing Graph-Neural Networks

Asad Aftab
Technische Universität Wien, Vienna, Austria
asad.aftab@tuwien.ac.at

***Abstract*—This project report explores the replication of Model Extraction Attacks specifically applied to Graph Attention Network (GATNet) models used for link prediction, extending the pioneering work on security vulnerabilities within Graph Neural Networks (GNNs). Initially, the focus was on broad GNN models; our study narrows this to GATNet, examining its susceptibility to sophisticated extraction methods. We adopted the structured approach originally proposed to categorize potential threats into seven distinct classes, depending on the attacker's knowledge, including node attributes and neighboring connections. Our replication involved implementing these established attack methods and evaluating their effectiveness on three real-world datasets specifically for link prediction tasks. The outcomes strongly align with the original study, demonstrating a successful model duplication in 84% to 92% of the cases. These results not only validate the initial attack methodologies but also highlight the critical need for robust security measures in GNN architectures used for link prediction.**

***Index Terms*—GNNs, Graph Attention Network (GATNet), Link Prediction**

I. Introduction

Graphs, with their inherent capability to model complex relationships and interactions among various entities, are extensively used across multiple domains such as social networks, communication networks, and biological networks. With the growing application of graph-based data in fields like machine learning and network security, understanding vulnerabilities and potential attack vectors against these data structures is crucial.

- **Structural Attacks**: These attacks target the integrity of the graph by manipulating its topology through the addition or removal of nodes and edges, potentially degrading the performance of algorithms that depend on the graph structure.
- **Attribute Attacks**: Focus on modifying the attributes of nodes or edges, potentially misleading algorithms reliant on these attributes for tasks like classification.
- **Inference Attacks**: Utilize graph data to infer private information about the network or its users, potentially re-identifying anonymized data.
- **Availability Attacks**: Aim to disrupt the availability of graph-based services, resembling denial-of-service attacks.
- **Manipulation Attacks**: Involve injecting false data or manipulating graph learning algorithms to achieve biased outcomes.
- **Evasion and Poisoning Attacks**: Occur when attackers modify input data during the inference stage or training data during the model training phase to mislead the learning process.

Among these attack types, Model Extraction Attacks are particularly challenging, especially for Graph Neural Networks (GNNs) like Graph Attention Network (GATNet), which are used for tasks such as link prediction. These attacks aim to reverse engineer a GATNet model by systematically querying it and reconstructing a similar model from the responses. This exploitation of MLaaS (Machine Learning as a Service) reveals significant vulnerabilities in terms of model confidentiality and integrity.

This work is a replication effort of some attacks mentioned in [3].

II. Preliminaries

This section outlines the GATNet architecture used for link prediction and discusses its susceptibility to model extraction attacks.

A. Link Prediction

Link prediction involves predicting the presence of a link between two nodes in a graph based on their attributes and the structure of the graph. A GATNet model, denoted as $f_\theta(\cdot)$, leverages attention mechanisms to focus on important nodes and relationships, enhancing the prediction accuracy. The model outputs a probability p_{ij} that a link exists between nodes v_i and v_j, and is expected to approximate the true link label y_{ij}.

B. Graph Attention Networks

We employ a GATNet model, which utilizes a self-attention strategy to compute the hidden representations of each node in the graph. The operation of GATNet for link prediction can be described by:

$$f_\theta(A, X) = \text{sigmoid}\left(\sum_k \alpha_{ijk}\text{LeakyReLU}(A \cdot X \cdot W^{(k)})\right),$$

where α_{ijk} are the attention coefficients that indicate the importance of node j's features to node i, A is the adjacency matrix, X are the node features, and $W^{(k)}$ represents the weight matrix for the k-th attention head.

III. Attack Setting

A. Adversary's Goal

In an MLaaS environment, GATNet models deployed for link prediction can be targeted by adversaries through model extraction attacks. The aim is to reconstruct a model that

performs similarly to the GATNet by leveraging observed input-output pairs from queries submitted via the service's API.

B. Adversarial Knowledge

The level of knowledge available to the attacker can vary, but in the context of GATNet and link prediction, we assume a black-box attack setting where the attacker only has access to a subset of adjacency and attribute information:

1) **Node Attributes (X) of the Target Training Graph**: Limited to known attributes of directly queried nodes.
2) **Graph Structure (A) of the Target Training Graph**: Knowledge may include observed edges from the subset of nodes available to the attacker.
3) **Shadow Dataset** $G' = (V', E', X')$: A similar dataset in the same domain, enabling the attacker to model and train a surrogate GATNet.

These structured settings facilitate the understanding and execution of model extraction attacks on GATNet models used for link prediction, highlighting the necessity for robust security measures to protect these advanced machine learning systems.

C. Attack Taxonomy

In assessing the security of GATNet models for link prediction, we categorize potential attacks based on the knowledge accessible to the attacker. This is denoted by the tuple (X, A, G'). We consider seven distinct scenarios based on the attacker's knowledge of these dimensions, excluding the case where the attacker lacks knowledge of all three dimensions.

- **Attack-0 (X, A, A_{k-hop})**: In this scenario, the attacker has access to the attributes of the attack nodes (X_A) and their local sub-graph structures ($A_{A,k-hop}$), including the connections up to a certain distance (k-hop) from the attack nodes. This knowledge allows the attacker to acquire a comprehensive understanding of the local neighborhood of the nodes. The attacker can utilize this information to gather labels and train a duplicate model. For instance, in a social network, an attacker might have partial access to user profiles and their immediate connections, which can be used to infer network patterns and train a surrogate model.
- **Attack-1 (X, *)**: This attack assumes the attacker only has access to the node attributes (X_A) of the attack nodes, with no information on the connections (A) or a shadow dataset. In this case, the attacker must rely solely on the attributes to gain insights. An example scenario might be an attacker with access to user profile information but without access to transactional or relational data. The attacker may use methods such as synthesizing connections or other inferential techniques to approximate the network structure based on the attributes.
- **Attack-2 (A, *)**: The attacker has access only to the graph structure (A) but lacks node attributes (X) and a shadow dataset. In such scenarios, like in certain social networks where connections between users are public but profile details are private, the attacker can exploit the available structural data. The attacker focuses on leveraging these connections to infer information about the network, potentially reconstructing parts of the graph or understanding the network's connectivity patterns.
- **Attack-3 (A, X, *)**: This scenario provides the attacker with full knowledge of the graph structure (A) but no access to node attributes (X). The attacker synthesizes attributes for each node using techniques like one-hot encoding, where each node's position is encoded uniquely. This allows the attacker to maintain the structural integrity necessary for tasks like link prediction. The primary challenge here is to determine if the synthesized attributes can effectively substitute for the real attributes in making accurate predictions. The attacker aims to assess the vulnerability of the model when relying primarily on structural information.

These structured attack scenarios aid in understanding the range of threats faced by GATNet models used for link prediction, highlighting the necessity for robust security measures that can adapt to various levels of adversarial knowledge.

Algorithm 1 Attack 0

Require:
dataset_name (`string`): {'AmazonCoPurchase'}
attack_node_arg (`float`): Proportion of nodes involved in the attack
cuda (`bool`): Indicates whether CUDA is available for use
Ensure:
Predicted link probabilities $\mathbf{P}$
Load data using **dataset_name**, obtaining $\mathbf{G} = (V, E)$
Initialize GATNet f_θ with parameters suitable for the graph size and features
$\mathbf{X} \leftarrow$ node features, $\mathbf{E} \leftarrow$ edges of $\mathbf{G}$
Prepare positive and negative edge samples $\mathbf{E}^+$ and $\mathbf{E}^-$
procedure GENERATE EDGE FEATURES
  for each edge $(i, j) \in \mathbf{E}^+ \cup \mathbf{E}^-$ **do**
    \\Prepare edge features
    $\mathbf{X}_{ij} \leftarrow$ concatenate$(\mathbf{X}_i, \mathbf{X}_j)$
  end for
end procedure
procedure TRAIN GATNET
  for `epoch` $= 1$ to N **do**
    \\Node embeddings
    $\mathbf{Z} \leftarrow f_\theta(\mathbf{X}, \mathbf{G})$
    for each $(i, j) \in \mathbf{E}^+ \cup \mathbf{E}^-$ **do**
      \\Predict link probability
      $s_{ij} \leftarrow \sigma(\mathbf{Z}_i^\top \mathbf{Z}_j)$
      Compute loss $\mathcal{L}$ for (i, j) based on s_{ij} and label y_{ij}
    end for
    Update θ using gradient descent on $\mathcal{L}$
  end for
end procedure
procedure EVALUATE MODEL
  Compute ROC-AUC and RMSE for $\mathbf{P}$ against true labels $\mathbf{Y}$
end procedure
Output $\mathbf{P}$ and performance metrics

IV. Attack Realization

A. Attack-0

We begin with a scenario where the attacker acquires a subset of nodes V_A from the graph and gains access to their attributes X_A and their local sub-graph structures $A_{A,k-hop}$. This reflects realistic scenarios where any node might potentially be an attack node.

To effectively replicate the target model, the attacker constructs what we refer to as an 'attack graph'. This graph incorporates node attributes, graph structure, and node labels derived from adversarial knowledge:

1) **Issuing Queries and Obtaining Labels:** The attacker uses known attributes and the responses from queries to label the attack nodes, employing these labels and attributes to train a duplicate model for node classification.
2) **Gathering Neighbour Connections:** The attacker collects information about connections between the attack nodes and their neighbors to maintain the influence of local graph topology on the node classification accuracy.
3) **Synthesising Attributes for Inaccessible Nodes:** If some neighbor nodes' attributes are unknown, the attacker synthesizes these attributes using a weighted combination of the known attributes of neighboring nodes, adjusted for node connectivity (degree).

These steps culminate in training a node classification GNN model through semi-supervised learning. The detailed steps are summarized in Algorithm 1 in the main text.

B. Attack-1

This attack intensifies the constraints on the attacker's knowledge, limiting it to only the attributes of the attack nodes X_A. In this scenario:

1) **Issuing Queries and Obtaining Labels:** As in Attack-0, attributes and query responses are used to label nodes within the attack graph.
2) **Synthesising Connections Among Attack Nodes:** Without knowledge of the graph structure, the attacker must infer or construct a substitute graph. Using attributes, the attacker employs a graph generation method, such as Learning Discrete Structures (LDS), to synthesize connections among nodes. This method focuses on optimizing the performance of the classification tasks.

After establishing a substitute graph with synthesized connections, the attacker proceeds to train the duplicated model using supervised learning. This approach avoids the isolation of nodes by approximating edge distributions and ensuring a density close to the target graph.

C. Attack-2

This attack scenario operates under the constraint that the attacker has access only to the attributes of the attack nodes X_A. The steps involved are:

1) **Issuing Queries and Obtaining Labels:** The attacker selects a set of attack nodes from the graph and uses a query mechanism to obtain labels for these nodes. The selection process ensures that the nodes are within valid bounds, and labels are retrieved using the available features.
2) **Extracting a Subgraph Based on Attack Nodes:** Without the complete graph structure, the attacker focuses on the subset of the graph containing the selected attack nodes. This subgraph is directly extracted from the dataset, ensuring that the number of nodes and the associated features are consistent. The subgraph's connectivity is retained from the original dataset, without additional structure synthesis.
3) **Generating Edge Samples for Training and Testing:** The attacker creates a training set for link prediction by identifying existing edges (positive samples) and generating non-edges (negative samples) within the subgraph. The edge labels are constructed accordingly, and the data is split into training and testing sets.
4) **Training the Attacked Model:** A GAT model is trained on the extracted subgraph. The model training involves optimizing for the binary classification task of predicting whether a pair of nodes is connected. This is done

Algorithm 2 Attack 1

Require: D: Dataset name, attack_node_arg: Attack node count, CUDA: CUDA flag
Ensure: Trained model parameters
procedure SetupLabels(D)
  $G, \mathbf{X}, \mathbf{Y} \leftarrow$ Load graph data
  $G \leftarrow$ Setup train, val, test masks
  return $G, \mathbf{X}, \mathbf{Y}$
end procedure
procedure TrainBaseline($G, \mathbf{X}, \mathbf{Y}$)
  Initialize f_θ with GATLinkPredictor
  for epoch $= 1$ to N **do**
    $\mathbf{Z} \leftarrow f_\theta(G, \mathbf{X})$
    $\mathcal{L} \leftarrow$ Compute loss with labels
    Update model f_θ using optimizer
  end for
  return f_θ
end procedure
procedure GenerateLDSGraph($\mathbf{X}$)
  similarity_matrix $\leftarrow$ Cosine similarity
  adj_matrix $\leftarrow$ Apply threshold
  return adj_matrix
end procedure
procedure Attack($G, \mathbf{X}$, attack_nodes)
  $\mathbf{X}_{\text{attack}} \leftarrow \mathbf{X}$[attack_nodes]
  $G_{\text{shadow}} \leftarrow$ Generate LDS graph for $\mathbf{X}_{\text{attack}}$
  Train attacked model on $G_{\text{shadow}}, \mathbf{X}_{\text{attack}}$
end procedure
procedure TrainAttackedModel($G_{\text{shadow}}, \mathbf{X}_{\text{attack}}$)
  Initialize $f_{\theta'}$ with GATLinkPredictor
  for epoch $= 1$ to n **do**
    $\mathbf{Z} \leftarrow f_{\theta'}(G_{\text{shadow}}, \mathbf{X}_{\text{attack}})$
    $\mathcal{L} \leftarrow$ Compute loss with labels
    Update model $f_{\theta'}$ using optimizer
  end for
  Evaluate fidelity with f_θ and $f_{\theta'}$
end procedure
return $f_{\theta'}$

Algorithm 3 Attack 2

Require: D: Dataset name, α: Attack fraction, CUDA: CUDA flag
Ensure: Trained model and metrics

procedure LOADANDPREPAREDATA(D)
  $G, \mathbf{X} \leftarrow$ LOADDATASET(D)
  $\mathbf{X} \leftarrow$ PADFEATURES($\mathbf{X}, G$)
  $G \leftarrow$ UPDATEGRAPH($G, \mathbf{X}$)
end procedure
procedure TRAINBASELINE($G, \mathbf{X}$)
  $f_\theta \leftarrow$ Init GATLinkPredictor()
  opt $\leftarrow$ Adam(f_θ)
  $\mathbf{E}, \mathbf{E}^-, \mathbf{L} \leftarrow$ GENEDGESAMPLES(G)
  for epoch $= 1 \rightarrow N$ **do**
    $\mathbf{Z} \leftarrow f_\theta(G, \mathbf{X})$
    $\mathcal{L} \leftarrow$ COMPUTELOSS($\mathbf{Z}, \mathbf{E}, \mathbf{E}^-, \mathbf{L}$)
    BACKPROP($\mathcal{L}$, opt)
    EVALUATE($f_\theta, G, \mathbf{X}$)
  end for
end procedure
procedure SIMATTACK($G, \mathbf{X}, \alpha$)
  $\mathbf{N}_{\text{attack}} \leftarrow$ SELECTNODES(G, α)
  $G_{\text{shadow}}, \mathbf{X}_{\text{shadow}} \leftarrow$ EXTRACTSUBGRAPH($G, \mathbf{N}_{\text{attack}}$)
  TRAINATTACK($G_{\text{shadow}}, \mathbf{X}_{\text{shadow}}$)
end procedure
procedure TRAINATTACK($G_{\text{shadow}}, \mathbf{X}_{\text{shadow}}$)
  $f_{\theta'} \leftarrow$ Init GATLinkPredictor()
  opt $\leftarrow$ Adam($f_{\theta'}$)
  $\mathbf{E}, \mathbf{E}^-, \mathbf{L} \leftarrow$ GENEDGESAMPLES(G_{shadow})
  for epoch $= 1 \rightarrow N$ **do**
    $\mathbf{Z} \leftarrow f_{\theta'}(G_{\text{shadow}}, \mathbf{X}_{\text{shadow}})$
    $\mathcal{L} \leftarrow$ COMPUTELOSS($\mathbf{Z}, \mathbf{E}, \mathbf{E}^-, \mathbf{L}$)
    BACKPROP($\mathcal{L}$, opt)
    EVALUATE($f_{\theta'}, G_{\text{shadow}}, \mathbf{X}_{\text{shadow}}$)
  end for
  EVALUATEFID($f_\theta, f_{\theta'}, G_{\text{shadow}}, \mathbf{X}_{\text{shadow}}, \mathbf{N}_{\text{attack}}$)
end procedure

by computing the logits for node pairs and using a binary cross-entropy loss function. The model is updated iteratively, and performance is monitored on the test set.

This approach does not involve generating a synthetic graph structure based on the attributes alone but rather leverages the existing subgraph connectivity from the original data. The attack's effectiveness is gauged by the fidelity metric, which assesses how well the attacked model mimics the baseline model's outputs.

D. Attack-3

In this attack scenario, the attacker has full knowledge of the graph structure A but lacks access to node attributes X for the vertex set V. The primary challenge is synthesizing node attributes and leveraging the graph structure for conducting a link prediction attack using a GATNet.

Steps of the Attack

1) **Graph Structure Utilization:**
   - The attacker utilizes the complete known graph structure A, which includes all connections but no attribute information of any nodes.
2) **Attribute Synthesis for Nodes:**
   - To address the lack of node attribute data, the attacker synthesizes attributes using one-hot encoding based on node indices. For example, for n nodes, the attribute for node v_i is a vector of size n with a 1 at the i-th position and 0s elsewhere.
   - This synthetic attribute setup ensures each node is uniquely identifiable by its position, which is critical for maintaining structural integrity in the absence of real attribute data.
3) **Model Training Strategy:**
   - The synthesized attributes and the graph structure are used to train a surrogate GATNet model. This model adopts a semi-supervised learning framework where only a subset of node links are labeled.
   - The training focuses on predicting the existence of links between node pairs, leveraging the structural information provided by the synthetic attributes and the complete graph knowledge.
4) **Objective of the Attack:**
   - The attack aims to replicate the target model's ability to predict links, thereby testing the model's robustness against an attack where only structural information is available.
   - This approach also evaluates whether a model trained on synthetic attributes can approximate the predictions of the original model, highlighting potential vulnerabilities in the model's reliance on structure over node-specific attributes.
5) **Challenges and Considerations:**
   - A key challenge is determining whether synthetic attributes can effectively replace real attributes in making meaningful link predictions. This issue is crucial because synthetic attributes may not capture complex attribute-based interactions typical in real scenarios.
   - The computational complexity of training a GATNet on a large graph with high-dimensional one-hot encoded attributes also needs to be considered.

V. EXPERIMENTS

In this section, we present a detailed set of experiments designed to evaluate our attack methodologies. We start by outlining the experimental setup and proceed to discuss the results of each attack.

A. Experimental Setup

Datasets. The primary datasets used in our evaluation are the AmazonCoBuyPhoto [2] and AmazonCoPurchaseBooks [1] datasets. These datasets consist of co-purchase networks where nodes represent products, and edges denote co-purchase relationships. The detailed statistics are provided in Table I.

Dataset Configuration. The datasets are configured differently based on the attack type. For **Attack-0, Attack-1, Attack-2**, and **Attack-3**, which do not assume knowledge of a shadow dataset, the entire graph is used to train the target

Algorithm 4 Attack 3

Require: D: Dataset, α: Attack fraction, CUDA: CUDA flag
Ensure: Trained model and metrics
$G, \mathbf{X}, \mathbf{Y} \leftarrow \text{LOADGRAPHDATA}(D)$
$G \leftarrow \text{ADDSELFLOOPS}(G)$
$\mathbf{M}_{\text{train}}, \mathbf{M}_{\text{val}}, \mathbf{M}_{\text{test}} \leftarrow \text{PARTITIONDATA}(G)$
$f_\theta \leftarrow$ Init GATNet()
procedure $\text{TRAINMODEL}(G, \mathbf{X}, \mathbf{Y}, \mathbf{M}_{\text{train}})$
  for epoch $= 1 \rightarrow N$ **do**
    $\mathbf{Z} \leftarrow f_\theta(G, \mathbf{X})$
    $\mathcal{L} \leftarrow \text{COMPUTELOSS}(\mathbf{Z}[\mathbf{M}_{\text{train}}], \mathbf{Y}[\mathbf{M}_{\text{train}}])$
    $\text{OPTIMIZE}(\mathcal{L})$
  end for
end procedure
procedure $\text{SIMATTACK}(G, \mathbf{X}, \alpha)$
  $\mathbf{N}_{\text{attack}} \leftarrow \text{SELECTNODES}(G, \alpha)$
  $\mathbf{X}' \leftarrow \text{MODIFYFEATURES}(\mathbf{X}, \mathbf{N}_{\text{attack}})$
  $f_{\theta'} \leftarrow$ Reinit GATNet()
  for epoch $= 1 \rightarrow N$ **do**
    Retrain $f_{\theta'}$ on $(G, \mathbf{X}')$
    Evaluate $f_{\theta'}$
  end for
end procedure

models. The dataset is split into 5% of the nodes for training, 11% for validation, and the remaining for testing. Specifically, we select 380 (5%) labeled nodes for training, 840 (11%) for validation, and 4,610 (60%) unlabeled nodes for testing.

Evaluation Metrics. We assess the performance of our attacks using two primary metrics: *fidelity* and *accuracy*. Fidelity measures the similarity between the surrogate and target models' outputs, defined as the fraction of links e_{ij} for which the predictions $f_{\theta'}(e_{ij}) = f_\theta(e_{ij})$. High fidelity indicates that the surrogate model closely mimics the target model's behavior, which is crucial for further analyses such as adversarial link prediction. Accuracy evaluates how correctly the surrogate model predicts the presence or absence of links, calculated as the proportion of correctly predicted links among the total possible links. Higher accuracy suggests that the surrogate model can be effectively used for link prediction tasks without additional queries to the target model, thereby posing a risk to the model owners.

Models. The target model used in our experiments is a Graph Attention Network (GAT) with two layers. The model's architecture includes:

- **Input Layer:** The input features are passed through a GATConv layer, which has a specified number of input features (`in_feats`), hidden features (`h_feats`), and multiple attention heads (`num_heads`). This first layer computes the attention scores among nodes in the graph and produces multiple attention-based hidden representations for each node.
- **Hidden Layer:** The outputs from the attention heads are concatenated and then passed through an Exponential Linear Unit (ELU) activation function. A dropout layer with a rate specified by the `dropout` parameter is applied after the activation to prevent overfitting.
- **Output Layer:** The processed hidden representations are fed into a second GATConv layer, which aggregates the information from multiple heads into a single output per node. The final output for each node is obtained by averaging the results from the attention heads (`mean(1)`).

The model's layers and activation functions are designed to capture and utilize the structural information present in the graph, as well as the features of the nodes. The inclusion of dropout helps to improve the generalization of the model by randomly dropping units during training.

VI. Attack Performance

A. Overview

The comprehensive analysis of our six attack configurations reveals notable differences in accuracy and fidelity. For Attack 0, Attack 1, and Attack 3, the proportion of attack nodes is varied from 5% to 25% of the total nodes. Meanwhile, Attack 0 with GraphSAGE and the corresponding variations with different values of alpha also illustrate distinct outcomes. The results, as depicted in the figures and tables, indicate that most of our attacks achieve accuracy and fidelity levels close to those of the target models, highlighting the efficacy of our attack strategies.

B. Attack 0

Attack 0 demonstrates a consistently high fidelity across various configurations, attributed to the substantial overlap in training data with the target model. Figure 1d illustrates the correlation between the number of attack nodes and the resulting accuracy and fidelity. Notably, as the proportion of attack nodes increases, both metrics improve, with accuracy peaking at 97.03% and fidelity at 94.049%. This trend underscores the effectiveness of utilizing a larger dataset fraction to achieve closer approximations to the target model, as evidenced by progressively higher fidelity and accuracy with an increase in attack nodes from 5% to 25%.

C. Attack 1

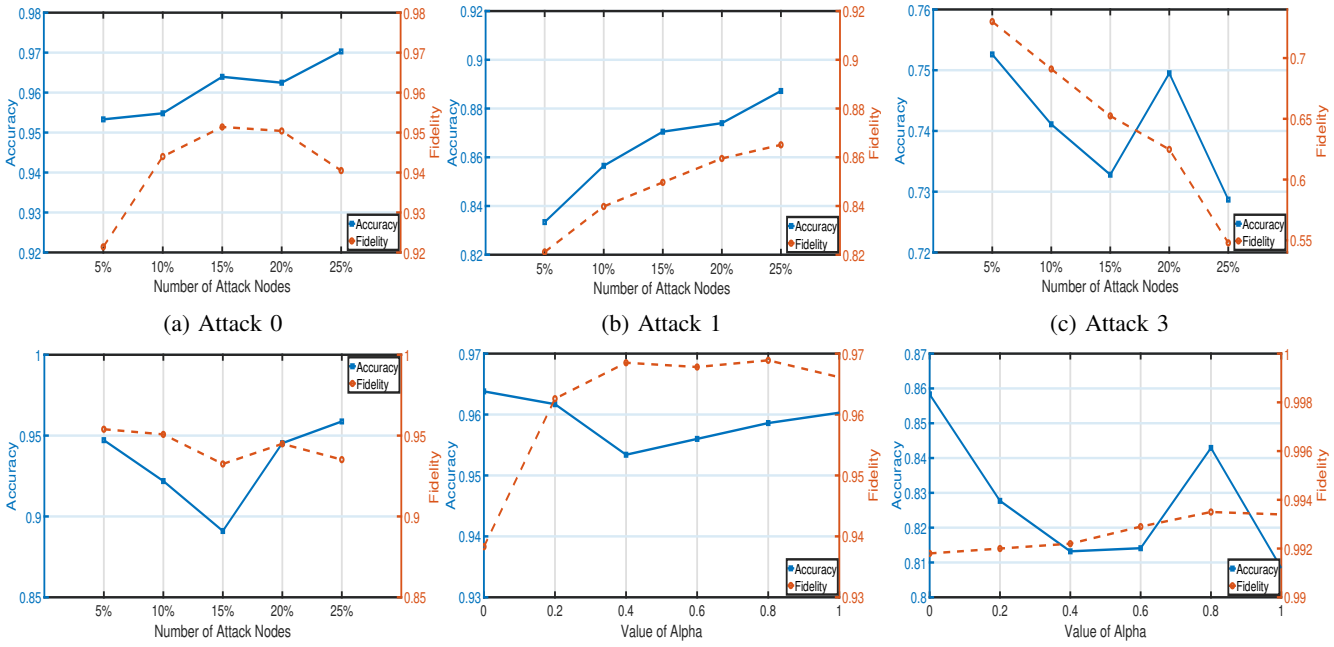
In Attack 1, depicted in Figure 1e, the attack model only has access to node attributes, with unknown connections. The results indicate a notable increase in both accuracy and fidelity with the addition of more attack nodes, reaching up to 89.65% accuracy and 86.98% fidelity at 20% attack nodes. However, an observed decrease at 25% attack nodes suggests potential overfitting or diminishing returns, indicating an optimal range for attack node selection.

D. Attack 3

As shown in Figure 1f, Attack 3, where only the graph structure is known, demonstrates the least effectiveness. The maximum accuracy and fidelity observed were 75.26% and

TABLE I: Statistics of the Dataset

Dataset	Nodes	Edges
AmazonCoBuyPhoto	7,650	238,163
AmazonCoPurchaseBooks	393,558	741,107



(a) Attack 0 (b) Attack 1 (c) Attack 3

(d) Attack 0: GraphSAGE Model (e) Attack 0: GAT Model (f) Attack 0: GraphSAGE Model

Fig. 1: Impact of the number of attack nodes and the adjustment of the factor α for Attack-0, Attack-1, and Attack-3.

73%, respectively, at 5% attack nodes. The declining trend in performance as the percentage of attack nodes increases to 25% highlights the challenges posed by limited attribute information, emphasizing the difficulty in accurately reconstructing the target model with insufficient data.

E. Attack 0: GraphSAGE Model

When employing the GraphSAGE model in Attack 0, the results, as illustrated in Figure 1d, show high accuracy and fidelity, with the best performance noted at 25% attack nodes, achieving 95.87% accuracy and 95.31% fidelity. This indicates that the GraphSAGE model is particularly adept at generalizing from partial data, providing a robust surrogate model performance.

F. Attack 0: GAT Model with Alpha Variation

The impact of varying the alpha parameter in the GAT model for Attack 0 is detailed in Figure 1e. The accuracy remains high, with a peak at 0.9638, while fidelity consistently exceeds 93%. The results suggest that adjusting alpha, which modulates the influence of node attributes and structure, can fine-tune the model's performance. Notably, an alpha of 0.8 achieves the best balance, with 95.86% accuracy and 96.89% fidelity, suggesting a balanced consideration of neighborhood information yields the most accurate surrogate model.

G. Attack 0: GraphSAGE Model with Alpha Variation

In the case of the GraphSAGE model with alpha variation, depicted in Figure 1f, the fidelity remains exceptionally high, close to 99% across all alpha values. The accuracy, however, fluctuates, suggesting that while the model faithfully replicates the target's outputs, the exact match to the target's decision boundaries varies with alpha adjustments. The highest accuracy observed is 85.83% at an alpha of 0, indicating a baseline preference without additional neighborhood aggregation.

VII. Conclusion

This work has systematically explored the susceptibility of Graph Neural Networks (GNNs), specifically GAT and GraphSAGE models, to model extraction attacks using the AmazonCoBuyPhoto and AmazonCoPurchaseBooks datasets. Our findings reveal that these models can be effectively duplicated, achieving high fidelity and accuracy, especially when attackers have access to comprehensive information such as node attributes and local sub-graph structures.

Key observations include the significant impact of the proportion of attack nodes and the influence of the alpha parameter on the accuracy and fidelity of the surrogate models. Notably, Attack 0 exhibited the highest performance metrics, demonstrating the critical role of accessible training data in model reconstruction. Conversely, scenarios with limited information, such as Attack 3, highlighted the challenges in achieving high fidelity, thus underscoring the vulnerabilities inherent in GNNs under constrained conditions.

References

[1] https://snap.stanford.edu/data/amazon-meta.html
[2] https://docs.dgl.ai/en/0.9.x/generated/dgl.data.AmazonCoBuyPhotoDataset.html
[3] Bang Wu, Xiangwen Yang, Shirui Pan, and Xingliang Yuan. Model Extraction Attacks on Graph Neural Networks: Taxonomy and Realisation