GRAIN: EXACT GRAPH RECONSTRUCTION FROM GRADIENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated learning allows multiple parties to train collaboratively while only Federated learning allows multiple parties to train collaboratively while only sharing gradient updates. However, recent work has shown that it is possible to exactly reconstruct private data such as text and images from gradients for both fully connected and transformer layers in the honest-but-curious setting. In this work, we present GRAIN, the first exact reconstruction attack on graph-structured data that recovers both the structure of the graph and the associated node features. Concretely, we focus on Graph Convolutional Networks (GCN), a powerful framework for learning on graphs. Our method first utilizes the low-rank structure of GCN layer updates to efficiently reconstruct and filter building blocks, which are subgraphs of the input graph. These building blocks are then joined to complete the input graph. Our experimental evaluation on molecular datasets shows that GRAIN can perfectly reconstruct up to 70% of all molecules, compared to at most 20% correctly positioned nodes and 32% recovered node features for the baseline.

1 INTRODUCTION

Graph Convolutional Networks (GCNs) have shown a great promise in learning on graph-structured data like social networks, traffic flows, molecules, as well as, healthcare and income data. Many of these applications, however, require large quantities of private data, which can be hard to collect due to privacy regulations and the reluctance of users to share their data due to fear of losing competitive advantage. This has naturally led to widespread use GCNs alongside Federated Learning (FL) which promises to protect the sensitive data of users (Xie et al., 2021; Zhang et al., 2021; Zhu et al., 2022; Lee et al., 2022; Lou et al., 2021; Peng et al., 2022).

However, the privacy of client data in FL for different domains including images (Zhang et al., 2023), text (Petrov et al., 2024), and tabular data (Vero et al., 2023) was recently shown to be severely violated by the introduction of gradient inversion attacks in the honest-but-curious setting (Zhu et al., 2019). In these attacks, the federated server infers the client data based on passively observed client gradients and the model weights on which they were computed. Unfortunately, no prior work has investigated the vulnerability of GCNs to such attacks.

This work: Gradient inversion attack on graphs In this work, we introduce the first gradient inversion attack on graphs called ***G**raph **R**econstruction **A**lgorithm for **In**version of Gradients* (**GRAIN**), specifically designed to attack Graph Convolutional Networks by reconstructing both, the graph structure and the node features. At the core of our method is an efficient filtering mechanism to correctly identify possible subgraphs, which are then pieced together to reconstruct the entire graph. In particular, we leverage span checks to exploit the rank-deficiency of GCN layer updates and recover the discrete set of node features at each layer, as well as the subgraph adjacency matrices. We then reconstruct the client input by applying a depth-first search-based (DFS-based) traversal algorithm to piece together the full graph from the recovered subgraphs.

We evaluate our attack on a real-world chemical dataset for molecule property prediction, where molecules are represented as graphs with nodes denoting atoms. We show that in the graph classification setting we can reconstruct up to 70% of all molecules exactly and achieve a reconstruction accuracy up to 85% when considering partial graph reconstructions. Further, even in the harder node classification setting we exactly reconstruct 66% of the molecules when node labels are known.

Main Contributions Our main contributions are:

- The first gradient inversion attack on Graph Neural Networks, recovering the graph structure and node features.
- An efficient filtering mechanism that correctly identifies client subgraphs as well as a graph reconstruction algorithm, building the input graph from the subgraphs.
- A new set of evalution metrics designed to measure the similarity between reconstructed graphs and the client data, enabling efficient evaluation of graph gradient inversion attacks.
- A thorough evaluation of GRAIN on real-world chemical datasets showing GCNs do not preserve client data privacy in realistic GCN applications.

We believe this work is a promising first step to understanding and quantifying the privacy risks associated with using graph data in federated learning.

2 Related Work

Gradient inversion attacks (Zhu et al., 2019), are attacks to Federated Learning that aim to infer the client's private data from the FL updates clients share with the federated server. As such, they assume knowledge of the updates themselves, as well as, the model weights on which the updates were computed. Depending on the attack model, gradient inversion attacks are either malicious (Boenisch et al., 2021; Fowl et al., 2022b;a; Chu et al., 2023; Wen et al., 2022) if the attacker can additionally manipulate the model weights sent to the clients, or honest-but-curious (Zhu et al., 2019; Phong et al., 2018; Zhao et al., 2020; Geiping et al., 2020; Geng et al., 2021; Zhang et al., 2023; Li et al., 2022; Deng et al., 2021; Balunovic et al., 2022; Dimitrov et al., 2024; Petrov et al., 2024; Vero et al., 2023) if the attack is executed passively by just observing model weights and updates.

In this work, we focus on the harder setting of honest-but-curious gradient inversion attacks. Most existing honest-but-curious attacks formulate gradient inversion as an optimization problem (Zhao et al., 2020; Geiping et al., 2020; Yin et al., 2021; Geng et al., 2021; Zhang et al., 2023; Li et al., 2022; Deng et al., 2021; Balunovic et al., 2022) where the attacker tries to obtain the data which corresponds to a client update that matches the observed one best. While, this approach is effective in many domains like images (Geiping et al., 2020; Yin et al., 2021; Geng et al., 2021; Zhang et al., 2023; Li et al., 2022) where the client data is continuous, it has been shown that the associated optimization problem is much harder to solve for domains where client inputs are discrete. Some prior works have attempted to alleviate this issue by relying on various continuous relaxation (Balunovic et al., 2022; Vero et al., 2023) to the discrete optimization problem with some success.

In contrast to such approaches, a recent line of work showed that gradient inversion can be solved exactly for both continuous (Dimitrov et al., 2024) and discrete inputs (Petrov et al., 2024) for certain neural network architectures. In particular, our work builds upon the work of Petrov et al. (2024), where the authors show that when there are very large but countable number of options for the client input data, one can exploit the low-rank structure of the gradient updates of fully connected layers to efficiently test all possibilities and keep only those that match the true input data. Similar to Petrov et al. (2024), our attack exploits the low-rankness of GCN layers resulting in exact reconstruction of both the graph structure and the node features. To the best of our knowledge we are the first attack specifically tackling this issue.

3 Background and Notation

In this section, we provide the background and notation necessary for understanding our method.

Graph Terminology First, we introduce the graph notation utilized in this work. For an undirected graph $\mathcal{G} = (V, E)$ with node set V of size $n = |V|$ and edge set E, we denote the degree of any node $v \in V$ with $\deg_{\mathcal{G}}(v)$. Further, for a pair of vertices $v_s, v_e \in V$, the distance $\text{dist}(v_s, v_e)$ denotes the number of edges in the shortest path connecting v_s to v_e. Finally, we introduce the notion of a degree-k neighborhood of a node v, defined by the subgraph $\mathcal{N}^k_{\mathcal{G}}(v) = (V^k_v, E^k_v) \subset \mathcal{G}$ consisting of all nodes $V^k_v = \{v' \in V \mid \text{dist}(v, v') \leq k\}$ in the graph at a distance $\leq k$ from v and the edges between them $E^k_v = \{e = (v_1, v_2) \in E \mid v_1 \in V(\mathcal{N}^{k-1}_{\mathcal{G}}(v)), v_2 \in V^k_v\}$ with $\mathcal{N}^0_{\mathcal{G}}(v) = \{v\}$.

Graph Convolutional Networks (GCNs) GCNs are a class of neural networks that operate on graph-structured data through a message-passing mechanism utilizing the graph edges E. In particular, the i^{th} GCN layer takes as an input a matrix $\boldsymbol{X}^i \in \mathbb{R}^{n \times d}$ of d-dimensional node features for each node in V and performs a combination of messages passing and non-linearity to produce the node features of the next layer $\boldsymbol{X}^{i+1}$:

$$\boldsymbol{X}^{i+1} = \sigma(\boldsymbol{Z}^i) = \sigma\left(\tilde{\boldsymbol{A}}\boldsymbol{X}^i\boldsymbol{W}^i\right), \tag{1}$$

where $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n \times n}$ is the normalized adjacency matrix, $\boldsymbol{W}^i \in \mathbb{R}^{d \times d'}$ is the weight matrix and σ is an activation function. The normalized adjacency is given by $\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$, where $\boldsymbol{A}$ is the adjacency matrix of the graph and $\boldsymbol{D}$ is the degree matrix. Further, we index the i-th layer input feature of a given node v as $\boldsymbol{X}_v^i$. In our case, we concatenate L GCN layers, which are then followed by a feed forward neural network to perform the readout. We denote by f_i for $i = 0, 1, 2, \ldots, L-1$ the function that maps the input graph to the output of the i^{th} GCN layer.

Gradient Filtering in Linear Layers Recently, Petrov et al. (2024) showed that one can leverage the gradients of the network loss $\mathcal{L}$ w.r.t. the weights $\boldsymbol{W}^i$ of the i^{th} linear layer $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^i}$ to search for the correct set of inputs $\boldsymbol{X}^i$ to the layer among a discrete set of possibilities via filtering enabled by the low-rankness of the weight updates. We restate the theoretical findings below:

Theorem 3.1. *If $n < d$ and if the matrix $\frac{\partial \mathcal{L}}{\partial \boldsymbol{Z}^i}$ is of full rank (rank n), then* $\operatorname{rowspan}(\boldsymbol{X}^i) = \operatorname{colspan}(\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^i})$.

To verify whether some input vector $\boldsymbol{z}$ could have been part of the client input, Petrov et al. (2024) performs a spancheck. Specifically, the distance between $\boldsymbol{z}$ and the subspace spanned by the column vectors of $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^i}$ is measured:

$$d(\boldsymbol{z}, \tfrac{\partial \mathcal{L}}{\partial \boldsymbol{W}^i}) \coloneqq \|\boldsymbol{z} - \operatorname{proj}(\boldsymbol{z}, \operatorname{colspan}(\tfrac{\partial \mathcal{L}}{\partial \boldsymbol{W}^i}))\|_2.$$

We assume $\boldsymbol{z}$ to have been part of the i-th layer input if $d(\boldsymbol{z}, \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^i}) < \tau$ for a chosen threshold τ.

4 Overview of GRAIN

In this section, we provide a high-level overview of our method GRAIN, a gradient inversion attack specifically designed to reconstruct graph-structured client training data in the Federated Learning setting assuming an honest-but-curious adversary. GRAIN has two phases.

In the first phase, we iteratively create a proposal set, $\mathcal{T}_l$, for each GCN layer l in the network. This set contains all degree-l building blocks, which are subgraphs of degree l.

We then *filter* out nodes that are incompatible with the gradient according to Lemma 5.1, which is based on Theorem 3.1 (Petrov et al., 2024). We finish the filtering by removing degree-L building blocks that do not pass a consistency check. Finally, using the elements of this set, we *reconstruct* the input graph via a tree-search-based approach.

Setting We apply GRAIN on the GCN architecture, where each GCN layer applies a single message passing operation between neighbors in the graph. The nodes are characterized by a set of f *discrete* features $\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2 \times \cdots \times \mathcal{F}_f$.

In Fig. 1 we present an overview of GRAIN which we now elaborate on.

Filtering We begin from a proposal set containing all possible nodes $\mathcal{T}_0$. Using Lemma 5.1 we remove all nodes that are not compatible with the gradient via a span check, resulting in $\mathcal{T}_0^* \subseteq \mathcal{T}_0$. These degree-0 building blocks can then be combined to one another to form degree-1 building blocks, constituting the set $\mathcal{T}_1$. As before, we can use Lemma 5.1 on the next GCN layer, dropping impossible degree-1 building blocks, resulting into $\mathcal{T}_1^* \subseteq \mathcal{T}_1$. This procedure can be extended to iteratively build degree-l building blocks $\mathcal{T}_l$ by gluing degree-1 building blocks $\mathcal{T}_1^*$, which can then be filtered as before to obtain $\mathcal{T}_l^*$. As a last step, we filter out building blocks that cannot be used to form a single larger building block, leaving us with the final set of degree-L building blocks $\mathcal{T}_B^*$.

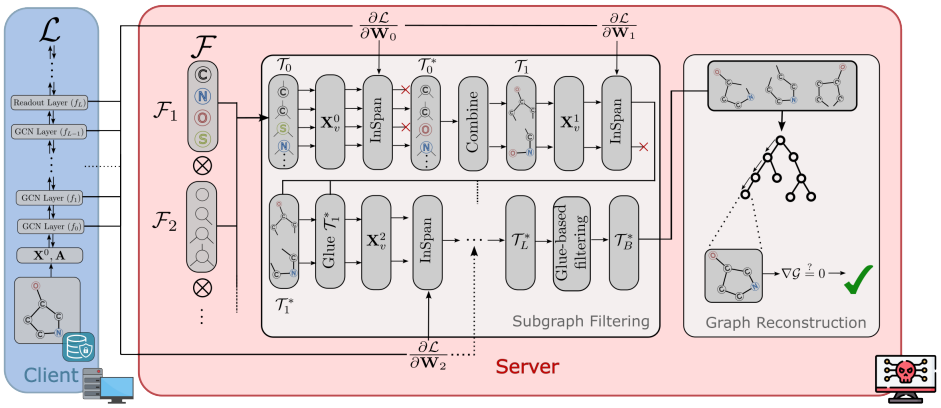


Figure 1: Overview of GRAIN. GRAIN first recovers the input nodes $\mathcal{T}_0^*$ by filtering through all possibilities $\mathcal{T}_0$. $\mathcal{T}_0$ is the cross-product of all possible feature values. Visualized are the sets of possible values for the atom type ($\mathcal{F}_1$) and the number of bonds ($\mathcal{F}_2$). It then iteratively combines and filters them to produce larger and larger building blocks up to a degree L. Finally, GRAIN reconstructs the input graph by combining building blocks from the filtered set $\mathcal{T}_B^*$ in a depth-first manner.

Graph Reconstruction Having arrived at a filtered set of degree-L building blocks, we iteratively glue together members of $\mathcal{T}_B^*$ at vertices that have not yet enough neighbors to match their feature degree. Here we leverage that the degree of a node is a widely used node feature for training GCNs (Hamilton et al., 2017; Xu et al., 2018; Cui et al., 2022), and is thus accessible by the attacker at this point. We explore all possible gluing options using a depth-first search. When we cannot extend the given graph further, we compute the gradients using it as an input and compare them to the observed gradients. If they do not match, we backtrack and try a different path. Otherwise, we can terminate our procedure successfully and return the reconstructed graph.

5 Subgraph Filtering and Graph Reconstruction

We now present the technical details of GRAIN. First, in Section 5.1, we describe the key operation of gluing two graphs together. Then, building on Theorem 3.1 (Petrov et al., 2024), we present Lemma 5.1 adapting Theorem 3.1 to GCN layers, allowing us to efficiently remove proposal elements of $\mathcal{T}_l$, which fail the span check and hence cannot be a subgraph of the input. Further, we detail how we construct $\mathcal{T}_B^*$ by utilizing consistency checks on $\mathcal{T}_L^*$. Finally, in Section 5.2 we propose the end-to-end graph reconstruction algorithm, which iteratively constructs the entire graph from the filtered set of possible subgraphs.

Algorithm 1 The GRAIN algorithm

function GRAIN($\mathcal{T}_0$, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}}$, τ, $\boldsymbol{f}$, Y)
  $\mathcal{T}_L^* \leftarrow$ CreateBBs($\mathcal{T}_0$, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}}$, τ, $\boldsymbol{f}$)
  $\mathcal{T}_B^* \leftarrow \{\}$
  for $\mathcal{G} \in \mathcal{T}_L^*$ **do**
    if $\Delta_{\mathcal{G}} == 0$ **then**
      return $\mathcal{G}$
    $CanGlue \leftarrow True$
    for $v \in V$ **do**
      if $!\exists \mathcal{G}^B \in \mathcal{T}_L^*$.glue$(\mathcal{G}, \mathcal{G}^B, v)$ **then**
        $CanGlue \leftarrow False$
        break
    if $CanGlue$ **then**
      $\mathcal{T}_B^* \leftarrow \mathcal{T}_B^* \cup \{\mathcal{G}\}$
  return ReconstructGraph($\mathcal{T}_B^*$, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}}$, Y)

5.1 Efficient filtering through spanchecks

We first describe the process of gluing a degree l building block $\mathcal{G}^B = (V^B, E^B)$ to a graph $\mathcal{G} = (V, E)$ at a vertex $v \in V$ resulting into $\hat{G} = \text{glue}(\mathcal{G}, \mathcal{G}^B, v)$ if v is the center of $\mathcal{G}^B$ and the degree-l neighborhood of v in $\mathcal{G}$ overlaps with $\mathcal{G}^B$, that is $\mathcal{N}_{\mathcal{G}}^l(v) \subseteq \mathcal{G}^B$. The resulting graph $\hat{G}$ is then the extension of the graph $\mathcal{G}$ by the building block $\mathcal{G}^B$ at vertex v, by attaching the non-overlapping

parts of $\mathcal{G}^B$ to $\mathcal{G}$ at v, as shown in Section 5.1. Formally, the result of the *gluing operation* is then $\hat{\mathcal{G}} = (V \cup V^B, E \cup E^B)$.

Span check for GCN layers We now state our main Lemma, building on Theorem 3.1 from Petrov et al. (2024) to be applied on GCNs. The proof can be found in Section A.

Lemma 5.1. *For $\frac{\partial \mathcal{L}}{\partial \boldsymbol{Z}^i}$ of full-rank, $d < n$, and a possibly normalized adjacency matrix at layer i, $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, $\boldsymbol{X}_j^i \in \text{colspan}(\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^i})$ if and only if $\boldsymbol{A}_j^T \notin \text{colspan}(\bar{\boldsymbol{A}}_j)$, where $\bar{\boldsymbol{A}}_j$ denotes the matrix $\boldsymbol{A}$ with its j-th column removed.*

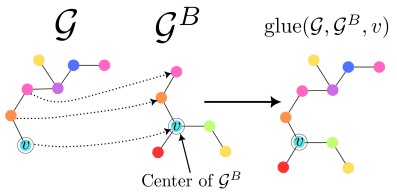


Figure 2: Visualization of the gluing operation.

Next, we begin our attack with the creation of the degree-L building blocks and reduce the search space via the span check mechanism. We start with the set of all possible nodes $\mathcal{T}_0$ and apply Lemma 5.1 to filter out the nodes that cannot be part of the input graph to get $\mathcal{T}_0^*$.

Creating degree-1 building blocks $\mathcal{T}_1^*$ We first define the extension $\text{ext}(v)$ of a node v to be the set of all possible graphs that can be constructed by attaching $\deg(v)$ nodes from $\mathcal{T}_0^*$ to v. We note, that we do not attach nodes $w \in \mathcal{T}_0^*$ to v if the feature degree of w is 0, that is $\deg(w) = 0$. The set of all possible degree-1 building blocks $\mathcal{T}_1 = \bigcup_{v \in \mathcal{T}_0^*} \text{ext}(v)$ is then defined as the set of all possible graphs that can be constructed by extending node from $\mathcal{T}_0^*$. We can then filter $\mathcal{T}_1$ by applying Lemma 5.1 on $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^1}$ to achieve the reduced set of degree-1 building blocks $\mathcal{T}_1^*$, as shown in Algorithm 2.

Creating degree-l building blocks $\mathcal{T}_l^*$ For a graph $\mathcal{G} \in \mathcal{T}_l^*$, we define the *dangling nodes* $\text{dang}(\mathcal{G})$ as the set of all nodes $v \in \mathcal{G}$ such that $\deg(v)$ is greater than the number of its neighbors. We extend the degree-l building blocks $\mathcal{G} \in \mathcal{T}_l^*$ by calculating all possible gluings of degree-1 building blocks $\mathcal{G}' \in \mathcal{T}_1^*$ to all dangling nodes of $\mathcal{G}$. This is shown in lines 9–15 of Algorithm 3. The resulting set is then called $\mathcal{T}_{l+1}$, which is then filtered by applying Lemma 5.1 on $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^{l+1}}$ to achieve the reduced set of degree-$l+1$ building blocks $\mathcal{T}_{l+1}^*$.

Algorithm 2 Filtering using the spancheck

```
1: function FILTER(T_l, ∂L/∂W^l, τ, f_{l-1})
2:     T_l^* ← {}
3:     for G in T_l do
4:         v ← center(G)
5:         if d(f_{l-1}(G)_v, ∂L/∂W^l) < τ then
6:             T_l^* ← T_l^* ∪ {G}
7:     return T_L^*
```

Additional structure-based filtering and likelihood ordering We repeat the process explained above until we reach the desired degree L. To further restrict the proposal set of building blocks, we perform a consistency check to further rule out building blocks that cannot be part of the ground truth graph. Specifically, we for every $\mathcal{G} \in \mathcal{T}_L^*$ and for every vertex $v \in \mathcal{G}$ there exist a building block in $\mathcal{T}_L^*$ that we can glue at v to $\mathcal{G}$. If this is not the case, we know that either $\mathcal{G}$ is the input graph, or it cannot be part of the ground truth graph and remove it from $\mathcal{T}_L^*$. The resulting set is then called $\mathcal{T}_B^*$. For small input graphs, it is possible that we already find the input graph during the creation of $\mathcal{T}_L^*$ and therefore cannot glue any building block to it. Thus, if we cannot glue any building block to a graph in $\mathcal{T}_L^*$, we check if the input graph could have generated the observed gradient using the gradient

Algorithm 3 Creating the degree-L building blocks

```
1: function CREATEBBS(T_0, ∂L/∂W, τ, f)
2:     T_0^*, T_1 ← FILTER(T_0, ∂L/∂W_0, τ, λv.X_v^0), {}
3:     for v in T_0^* do
4:         T_1 ← T_1 ∪ ext(v, T_0^*)
5:     T_1^* ←FILTER(T_1, ∂L/∂W_1, τ, f_0)
6:     for l ← 2, ..., L do
7:         T_l ← {}
8:         for G in T_{l-1}^* do
9:             S ← {G}
10:            for v in dang(G) do
11:                S' ← {}
12:                for G', G^B in S × T_1^* do
13:                    S' ← S' ∪ glue(G', G^B, v)
14:                S ← S'
15:            T_l ← T_l ∪ S
16:        T_l^* ← FILTER(T_l, ∂L/∂W_l, τ, f_{l-1})
17:    return T_L^*
```

distance $\Delta_{\mathcal{G}} = \min_{y \in Y} \| \frac{\partial \mathcal{L}(\mathcal{G}, y)}{\partial \boldsymbol{W}} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} \|_F$, where Y is the set of all possible labels and $\| \cdot \|_F$ is the standard Frobenius norm. If $\Delta_{\mathcal{G}} = 0$, GRAIN returns $\mathcal{G}$. This is shown in lines 3–13 of Algorithm 1.

5.2 Graph Reconstruction

We now take the filtered set of building blocks $\mathcal{T}_B^*$ and explore in a depth-first manner the set of graphs we can create by combining them. In order to speed up the search procedure, we order the building blocks in $\mathcal{T}_B^*$ by a score $S(\mathcal{G}^B)$. We first define the score $S_v(\mathcal{G}^B)$ to be equal to the lowest span check distance $d(\mathcal{G}, \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^L})$ of a building block $\mathcal{G}$ that can be glued to $\mathcal{G}^B$ at v. The score for the entire block is then calculated as the sum of the vertex scores $S(\mathcal{G}^B) = \sum_v S_v(\mathcal{G}^B)$.

Starting from the first block in the given order, we generate all possible graphs that can be created by gluing a building block $\mathcal{G}^B \in \mathcal{T}_B^*$ to a dangling node $v \in \mathcal{G}$ of the current graph $\mathcal{G}$, resulting in a new graph $\bar{\mathcal{G}} = \text{glue}(\mathcal{G}, \mathcal{G}^B, v)$. After every step, we enumerate all sets S of pairs (v_1, v_2) of vertices of $\bar{\mathcal{G}}$ such that the features of v_1 and v_2 match. For every such set S, we additionally consider for exploration the graph $\hat{\mathcal{G}} = \text{overlap}(\bar{\mathcal{G}}, S)$ created by overlapping each pair of vertices in S. Whenever the graph has no more dangling nodes, we compute $\Delta_{\mathcal{G}}$, successfully terminating if it is found to be 0, and keeping the graph with the lowest $\Delta_{\mathcal{G}}$ otherwise. The skeleton of the algorithm is seen in Algorithm 4, with each step detailed in Algorithm 5. The finalized GRAIN algorithm is then shown in Algorithm 1.

6 Evaluation

In this section we evaluate GRAIN's performance compared to prior work in the gradient leakage field. We introduce a new set of metrics, capturing the distance in node features and adjacency matrices. GRAIN shows significant improvements over existing attacks. Further, we show that GRAIN remains effective across a wide range of changes to the architecture.

Algorithm 4 Reconstructing the full graph

```
1: function ReconstructGraph(T*_B, ∂L/∂W, Y)
2:     G_best ← ∅
3:     d_best ← ∞
4:     S ← T*_B
5:     for B in T*_B do
6:         d_curr, G_curr ← DoDFS(T*_B, ∂L/∂W, B, Y)
7:         if d_curr = 0 then
8:             return 0, G_curr
9:         if d_curr < d_best then
10:            d_best, G_best ← d_curr, G_curr
11:    return d_best, G_best
```

6.1 Experimental setup

Next, we describe the architecture, introduce the datasets and describe 2 baseline algorithms we evaluate GRAIN against.

Architecture details We apply our attack on a 2-layer GCN ($L = 2$) with a hidden embedding dimension $d' = 300$ and a ReLU activation. The network also features a 2-layer feedforward network for performing the readout, as this is a common depth Kipf & Welling (2016). Given the depth restrictions, we recover building blocks up to a degree of 2, with the first readout layer being used for the relevant filtering of the largest blocks. Furthermore, in Table 3 we show that our attack is robust with respect to changes in the architecture parameters.

Data and datasets We evaluate on molecule property prediction data, where molecules are represented as graphs and each nodes represents an atom. We follow the common convention to omit hydrogen atoms in the graphs. Each node is embedded via concatenating the one-hot encodings of 8 different features (Xu et al., 2018; Wu et al., 2020), namely the atom type, formal charge, number of bonds, chirality, number of bonded hydrogen atoms, atomic mass, aromaticity and hybridization (Rong et al., 2020). We evaluate GRAIN on 100 samples of 3 well-known chemical datasets, namely Clintox, Tox21 and BBBP, introduced by the **MoleculeNet** benchmark (Wu et al., 2018).

Computational details We provide an efficient GPU imlementation, where each experiment has been run on a NVIDIA L4 Tensor Core GPU with less than 40GB of CPU memory.

6.2 BASELINE ATTACKS

We adapt the DLG attack (Zhu et al., 2019), a standard continuous attack, and TabLeak, an attack purposefully designed for recovering discrete tabular data. As described in (Vero et al., 2023), all input features are first passed through an initial sigmoid layer to ensure they are in the interval (0, 1). Similarly, we ensure the adjacency matrix A is symmetric by optimizing over the upper triangle, and apply a softmax operation over the dummy labels to convert them to probabilities. Finally, we generate a prediction graph by connecting all nodes v_i, v_j corresponding to $\sigma(A)_{ij} \geq 0.5$. Additionally, we test both baselines when they are given the correct adjacency matrix A. In all cases we provide the attack with the correct number of nodes to ensure that $\boldsymbol{X}$ and A have the correct shape. We demonstrate that, even when the baselines have a significant amount of prior knowledge, GRAIN significantly outperforms them (see Fig. 3 and Table 1).

6.3 EVALUATION METRIC

We introduce a set of metrics designed to evaluate the similarity of a pair of graphs $\mathcal{G} = (\mathcal{V}, \boldsymbol{A}, \boldsymbol{X})$, $\hat{\mathcal{G}} = (\hat{\mathcal{V}}, \hat{\boldsymbol{A}}, \hat{\boldsymbol{X}})$ under the common name GRAPH-N. We define GRAPH-N$_F(\mathcal{G}, \hat{\mathcal{G}})$ under a set of functions $F = \{F_k\}_{k=1}^{N}$, where for all k $F_k : \mathcal{G} \to \mathbb{R}^{|\mathcal{V}| \times d}$ is a function that aggregates the feature vectors for each k-degree neighborhood. This allows us to measure the similarites in features across increasingly larger subgraphs, which capture the structure around each node. In our case we utilise a randomly initialised $\geq k$-layer GCN to achieve such a mapping.

We note that a precise evaluation of the metric requires for us to match the 2 graphs as accurately as possible. Since exact matching of graphs is an NP-complete problem (Fortin, 1996), we match the graph nodes by applying the Hungarian matching algorithm (Frank, 2005) for minimizing a cost function $\boldsymbol{C}$ that captures the feature difference across degree-(0-5) neighborhoods:

$$\boldsymbol{C}_{ij} = \sum_{k=0}^{2} \sum_{m=1}^{d} (F_k(\mathcal{G}) - F_k(\hat{\mathcal{G}}))_m^2$$

We can hence define:

$$\text{GRAPH-N}_F(\mathcal{G}, \hat{\mathcal{G}}) = \begin{cases} \text{F1-Score}(F_N(\mathcal{G}), F_N(\hat{\mathcal{G}})) & \text{if } F_N(\mathcal{G}) \text{ - discrete} \\ R^2(F_N(\mathcal{G}), F_N(\hat{\mathcal{G}})) & \text{if } F_N(\mathcal{G}) \text{ - continuous} \end{cases}$$

We utilise 3 separate instances of the metric - namely for $N = 0, 1, 2$, where neighborhoods of higher degree are used to capture more structural information. It is important to note that all measurements are scaled by a factor of $\frac{\min(|\mathcal{V}|, |\hat{\mathcal{V}}|)}{\max(|\mathcal{V}|, |\hat{\mathcal{V}}|)}$ to penalize reconstructions of incorrect size. We further report the percentage of exactly reconstructed graphs for each method, denoted by FULL in the result tables.

6.4 EXPERIMENTAL RESULTS

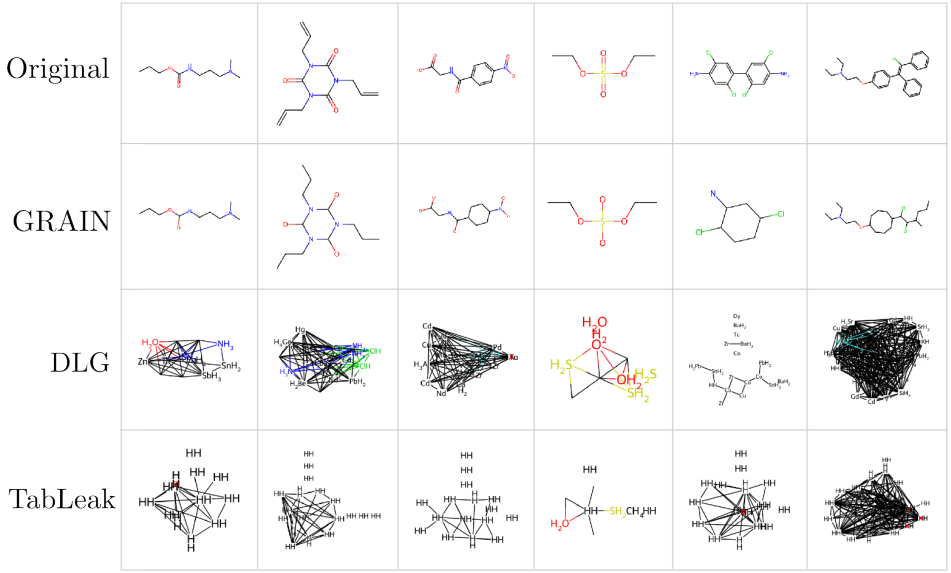
Next, we evaluate the baselines and GRAIN and show that GRAIN outperforms the existing baselines across all defined metrics. Further, GRAIN is applicable across a variety of settings, including being depth- and width-agnostic. In all measurements we quote the mean value of the metric, as well as the 95% confidence interval around it, measured by generating 10,000 random sample sets via bootstrapping.

Main experiments We first apply the algorithms DLG, TabLeak and GRAIN to the 3 datasets. We observe in Table 1 that GRAIN achieves a much higher partial reconstruction rate (between 70-85%) compared to any baseline. This remains true even when the baseline is informed about the input adjacency matrix A. When A is not given the metrics tend to decrease with neighborhood size, showing the baselines' inability to recover the structure. Not only do we observe much higher results on partial reconstruction, we also see we are able to recover between 35-70% of the dataset exactly, while the baselines can achieve this only in the case of very small molecules. For visual inspection, we also include a comparison of reconstructed molecules in Fig. 3. In this set of examples, the first

Table 1: Results (in %) of main experiments on 3 biochemical datasets – Tox21, Clintox, BBBP. Here "$+A$" refers to the baseline attack with the input adjacency matrix given.

		GRAPH-0	GRAPH-1	GRAPH-2	FULL
Tox21	GRAIN	$\mathbf{86.9}^{\mathbf{+4.2}}_{\mathbf{-5.7}}$	$\mathbf{83.9}^{\mathbf{+5.2}}_{\mathbf{-6.9}}$	$\mathbf{82.6}^{\mathbf{+5.7}}_{\mathbf{-7.4}}$	$\mathbf{68.0 \pm 1.7}$
	DLG	$31.8^{+4.5}_{-4.3}$	$20.3^{+5.5}_{-4.8}$	$22.8^{+6.6}_{-5.6}$	1.0 ± 0.2
	DLG $+A$	$54.7^{+3.9}_{-4.2}$	$60.1^{+4.6}_{-5.2}$	$76.7^{+3.6}_{-4.8}$	1.0 ± 0.2
	TabLeak	$25.1^{+5.1}_{-4.3}$	$12.4^{+5.5}_{-4.3}$	$10.8^{+5.6}_{-3.9}$	1.0 ± 0.2
	TabLeak $+A$	$55.6^{+3.9}_{-3.9}$	$57.7^{+4.1}_{-4.6}$	$73.8^{+2.8}_{-3.5}$	1.0 ± 0.2
Clintox	GRAIN	$\mathbf{73.7}^{\mathbf{+5.7}}_{\mathbf{-6.5}}$	$\mathbf{68.4}^{\mathbf{+6.7}}_{\mathbf{-7.8}}$	$66.8^{+7.0}_{-7.6}$	$\mathbf{36.0 \pm 1.2}$
	DLG	$24.0^{+4.1}_{-3.8}$	$10.3^{+4.8}_{-3.6}$	$12.2^{+5.5}_{-4.2}$	1.0 ± 0.2
	DLG $+A$	$52.5^{+3.2}_{-3.6}$	$52.6^{+4.1}_{-4.7}$	$\mathbf{72.3}^{\mathbf{+3.2}}_{\mathbf{-3.9}}$	1.0 ± 0.2
	TabLeak	$17.6^{+3.7}_{-2.8}$	$6.0^{+4.0}_{-2.4}$	$5.4^{+4.2}_{-2.5}$	1.0 ± 0.2
	TabLeak $+A$	$54.0^{+3.4}_{-3.3}$	$52.0^{+3.8}_{-4.2}$	$62.8^{+3.3}_{-4.2}$	1.0 ± 0.2
BBBP	GRAIN	$\mathbf{71.7}^{\mathbf{+5.9}}_{\mathbf{-6.8}}$	$\mathbf{66.8}^{\mathbf{+6.9}}_{\mathbf{-7.7}}$	$64.9^{+7.2}_{-8.0}$	$\mathbf{38.0 \pm 1.2}$
	DLG	$22.6^{+3.6}_{-3.3}$	$8.8^{+4.9}_{-3.2}$	$10.0^{+5.3}_{-3.7}$	0.0 ± 0.0
	DLG $+A$	$51.6^{+3.1}_{-3.6}$	$50.1^{+3.8}_{-4.5}$	$70.6^{+3.1}_{-4.2}$	0.0 ± 0.0
	TabLeak	$17.6^{+3.8}_{-2.8}$	$6.3^{+3.8}_{-2.5}$	$4.7^{+3.7}_{-2.3}$	0.0 ± 0.0
	TabLeak $+A$	$59.1^{+3.1}_{-3.6}$	$59.4^{+3.6}_{-4.3}$	$\mathbf{71.9}^{\mathbf{+2.9}}_{\mathbf{-4.0}}$	0.0 ± 0.0

Figure 3: Examples of molecule reconstructions. We note that both GRAIN and DLG do not recover the multivalent interactions, as this is an edge property that is not considered for GCNs.



4 columns show the exact reconstruction of the input. We also highlight that in cases where GRAIN does not managed to recover the entire graph, the attack can reconstruct subgraphs of the input (5th column), and a more realistic approximation otherwise (6th column).

Effect of graph size on reconstruction In Table 8 we show how GRAIN performs on molecules of different sizes. The molecules are divided into groups where the number of nodes satisfy $n \leq 15$, $16 \leq n \leq 25$ and $n \geq 26$ respectively, aggregated across all 3 datasets. We notice that GRAIN significantly outperforms the baselines for smaller graphs, but the performance decreases on the largest group. This limitation here stems from a time out (15 minutes) on the graph reconstruction

Table 2: Results (in %) of GRAIN and baselines in cases of different number of nodes n

	$n \leq 15$			$16 \leq n \leq 25$			$26 \leq n$		
	GRAPH-0	GRAPH-2	FULL	GRAPH-0	GRAPH-2	FULL	GRAPH-0	GRAPH-2	FULL
GRAIN	$\mathbf{93.0^{+3.4}_{5.4}}$	$\mathbf{91.6^{+3.8}_{6.3}}$	$\mathbf{81.9 \pm 1.7}$	$\mathbf{81.7^{+3.9}_{4.8}}$	$\mathbf{74.8^{+5.8}_{6.3}}$	$\mathbf{43.6 \pm 1.1}$	$50.1^{+6.8}_{7.1}$	$39.2^{+8.5}_{7.7}$	$\mathbf{5.1 \pm 0.6}$
DLG	$27.4^{+4.2}_{3.8}$	$13.3^{+5.7}_{4.6}$	1.0 ± 0.2	$25.5^{+3.9}_{3.5}$	$16.7^{+5.2}_{4.4}$	0.9 ± 0.2	$25.4^{+4.8}_{4.3}$	$14.8^{+6.4}_{5.3}$	0.0 ± 0.0
DLG $+A$	$52.1^{+3.1}_{3.3}$	$71.3^{+3.1}_{3.9}$	1.0 ± 0.2	$53.7^{+3.2}_{3.5}$	$75.3^{+3.0}_{3.7}$	0.9 ± 0.2	$53.0^{+4.3}_{4.8}$	$72.6^{+4.1}_{5.8}$	0.0 ± 0.0
TabLeak	$30.3^{+5.0}_{4.4}$	$15.4^{+5.8}_{4.8}$	1.9 ± 0.3	$15.7^{+3.0}_{2.2}$	$2.1^{+2.2}_{1.1}$	0.0 ± 0.0	$13.0^{+3.3}_{2.3}$	$2.8^{+4.1}_{1.9}$	0.0 ± 0.0
TabLeak $+A$	$53.9^{+4.0}_{4.2}$	$72.9^{+3.4}_{3.9}$	1.9 ± 0.3	$57.1^{+3.1}_{3.5}$	$71.4^{+2.9}_{3.6}$	0.0 ± 0.0	$\mathbf{56.1^{+2.9}_{3.3}}$	$\mathbf{74.4^{+2.3}_{3.1}}$	0.0 ± 0.0

Table 3: Results (in %) of GRAIN and the baselines in cases of different model parameters. Here L is the number of GCN layers and d' is the model's width. $L = 2, d' = 300$ is the original setting

		GRAPH-0	GRAPH-1	GRAPH-2	FULL
$L = 2$, $d' = 300$ (default)	GRAIN	$\mathbf{86.9^{+4.2}_{5.7}}$	$\mathbf{83.9^{+5.2}_{6.9}}$	$\mathbf{82.6^{+5.7}_{7.4}}$	$\mathbf{68.0 \pm 1.7}$
	DLG	$31.8^{+4.5}_{4.3}$	$20.3^{+5.5}_{4.8}$	$22.8^{+6.6}_{5.6}$	1.0 ± 0.2
	DLG $+A$	$54.7^{+3.9}_{4.2}$	$60.1^{+4.6}_{5.2}$	$76.7^{+3.6}_{4.8}$	1.0 ± 0.2
	TabLeak	$25.1^{+5.1}_{4.3}$	$12.4^{+5.5}_{4.3}$	$10.8^{+5.6}_{3.9}$	1.0 ± 0.2
	TabLeak $+A$	$55.6^{+3.9}_{3.9}$	$57.7^{+4.1}_{4.6}$	$73.8^{+2.8}_{3.5}$	1.0 ± 0.2
$L = 3$, $d' = 300$	GRAIN	$\mathbf{82.5^{+5.7}_{7.7}}$	$\mathbf{80.7^{+6.3}_{7.7}}$	$\mathbf{80.4^{+6.2}_{7.8}}$	$\mathbf{63.0 \pm 1.6}$
	DLG	$20.3^{+4.3}_{3.4}$	$7.8^{+5.1}_{3.3}$	$8.2^{+5.3}_{3.4}$	1.0 ± 0.2
	DLG $+A$	$43.0^{+3.7}_{3.6}$	$48.0^{+4.3}_{4.5}$	$66.0^{+3.7}_{4.6}$	1.0 ± 0.2
	TabLeak	$16.5^{+3.8}_{2.9}$	$8.8^{+4.4}_{3.1}$	$8.0^{+4.3}_{3.0}$	1.0 ± 0.2
	TabLeak $+A$	$47.5^{+4.0}_{4.2}$	$48.1^{+4.8}_{5.0}$	$62.9^{+4.3}_{4.4}$	1.0 ± 0.2
$L = 4$, $d' = 300$	GRAIN	$\mathbf{83.9^{+5.5}_{7.4}}$	$\mathbf{82.8^{+5.9}_{7.7}}$	$\mathbf{82.8^{+6.0}_{7.9}}$	$\mathbf{64.0 \pm 1.6}$
	DLG	$14.1^{+3.8}_{2.8}$	$4.0^{+4.7}_{2.2}$	$4.8^{+4.9}_{2.6}$	1.0 ± 0.2
	DLG $+A$	$39.1^{+3.7}_{3.8}$	$37.0^{+5.3}_{5.4}$	$55.6^{+5.0}_{5.7}$	1.0 ± 0.2
	TabLeak	$12.0^{+3.4}_{1.9}$	$2.1^{+4.3}_{1.4}$	$3.4^{+4.0}_{1.7}$	1.0 ± 0.2
	TabLeak $+A$	$30.0^{+4.7}_{4.0}$	$27.3^{+5.9}_{5.1}$	$51.1^{+4.9}_{5.3}$	1.0 ± 0.2
$L = 2$, $d' = 200$	GRAIN	$\mathbf{84.6^{+4.6}_{6.4}}$	$\mathbf{81.4^{+5.8}_{6.9}}$	$\mathbf{80.5^{+5.9}_{7.2}}$	$\mathbf{62.0 \pm 1.6}$
	DLG	$30.8^{+4.5}_{4.1}$	$18.9^{+5.8}_{4.9}$	$22.2^{+6.7}_{5.4}$	1.0 ± 0.2
	DLG $+A$	$50.3^{+4.2}_{4.2}$	$53.4^{+5.3}_{5.9}$	$68.7^{+4.9}_{6.1}$	3.0 ± 0.4
	TabLeak	$22.1^{+4.8}_{3.7}$	$10.3^{+5.3}_{3.6}$	$8.9^{+5.5}_{3.6}$	1.0 ± 0.2
	TabLeak $+A$	$55.0^{+4.8}_{5.0}$	$62.1^{+4.9}_{5.9}$	$76.7^{+3.6}_{4.7}$	1.0 ± 0.2
$L = 2$, $d' = 400$	GRAIN	$\mathbf{85.2^{+4.6}_{6.1}}$	$\mathbf{81.5^{+5.4}_{7.1}}$	$\mathbf{80.1^{+6.1}_{7.5}}$	$\mathbf{63.0 \pm 1.6}$
	DLG	$35.1^{+4.9}_{4.7}$	$26.1^{+6.4}_{5.6}$	$25.0^{+6.9}_{6.0}$	1.0 ± 0.2
	DLG $+A$	$57.6^{+3.9}_{4.3}$	$61.7^{+4.7}_{5.5}$	$72.5^{+4.3}_{5.5}$	2.0 ± 0.3
	TabLeak	$28.5^{+4.5}_{4.0}$	$17.1^{+5.4}_{4.4}$	$12.9^{+5.4}_{4.0}$	1.0 ± 0.2
	TabLeak $+A$	$61.7^{+3.6}_{3.7}$	$62.6^{+3.6}_{4.4}$	$76.3^{+2.9}_{3.3}$	1.0 ± 0.2

While this is a result of the computational limitations of our algorithm, the chemical setting is inherently difficult as is discussed in Section 7. Further, our work still manages to reconstruct a fraction of the large graphs exactly, which is impossible for the baseline models.

Effect of Model Parameters on Reconstruction Quality In Table 3 we demonstrate the performance of GRAIN under modifying the model parameters. We observe that neither changing in the number of layers nor the hidden dimension size of the GCN substantially affects the performance of GRAIN, while reaffirming the significant improvement over the baselines, even when they are given the graph connections as prior knowledge. We note that we only utilise the first 2 GCN layers even when $L > 2$, showing the robustness of our method.

Table 4: Results (in %) of GRAIN in the following cases: Row 1 – the original setting, Row 2 – the activation function GELU instead of ReLU, Row 3 – in the gradients shared are from a trained model instead of the first epoch, Row 4 – results of GRAIN in node classification task

	GRAPH-0	GRAPH-1	GRAPH-2	FULL
GRAIN (default)	$86.9^{+4.2}_{-5.7}$	$83.9^{+5.2}_{-6.9}$	$82.6^{+5.7}_{-7.4}$	68.0 ± 1.7
σ = GELU	$82.0^{+5.3}_{-6.7}$	$79.1^{+6.0}_{-7.4}$	$78.4^{+6.2}_{-8.0}$	61.0 ± 1.6
Pre-trained GCN	$73.5^{+6.4}_{-7.4}$	$70.0^{+7.3}_{-7.7}$	$68.6^{+7.6}_{-8.3}$	49.0 ± 1.4
Node classification	$88.0^{+3.8}_{-5.4}$	$85.5^{+4.6}_{-6.5}$	$84.9^{+5.0}_{-6.6}$	66.0 ± 1.6

Additional experiments We provide additional experiments showcasing GRAIN's performance in different miscellaneous settings in Table 10. First, we replace the ReLU activation function in the GCN by a GELU and report that GRAIN achieves similar results, showing our flexibility with respect to different activations. Furthermore, while prior work has shown that gradient inversion becomes significantly more difficult on pre-trained models (Geiping et al., 2020), GRAIN still manages to reconstruct around 50% of molecules exactly. Finally, we observe consistently good results when changing our task to a node classification one. We clarify that in this setting we additionally assume knowledge of the ground-truth labels, as they can be easily recovered with methods, such as the one described by Zhao et al. (2020).

7 Limitations

GRAIN is the first algorithm to make progress in the field of gradient inversion of GNN updates and, as such, we recognize significant potential for expanding upon our work further. Currently, our attack method is focused only on GCNs and depends on the assumption that the FL protocol uses the node degree as a node feature. While these assumptions apply to many GNN architectures, relaxing them is an important avenue for future work. Another key item for future work is reducing the computational complexity of GRAIN, to enable its scale to larger graphs. We believe this is a promising direction of research, as we believe that many further optimizations can be explored to improve the efficiency of our algorithm. Further, as described in Section 5, GRAIN requires that $n < d'$ to maintain the low-rank nature of the gradient updates. Although this is a limitation of our work, we believe that this assumption is satisfied for many practical settings, thereby exposing real client updates to considerable privacy risk. Finally, we leave the exploration of possible defenses against GRAIN to future work.

8 Conclusion

We introduced GRAIN, the first gradient inversion attack for Graph Neural Networks that is able to accurately recover graphs from gradients shared by the server. By leveraging the rank-deficiency of the GCN layers, we developed an efficient framework for extracting and filtering subgraphs of the input graph. We then presented an algorithm capable of reconstructing the original graph by iteratively combining the filtered subgraphs.

Our results showed GRAIN achieves an exact reconstruction rate up to 70% of the graphs in chemical datasets trained for graph classification. Additionally, we introduced new metrics to evaluate partial graph reconstructions and demonstrated that GRAIN significantly outperforms prior work. Finally, we showed that GRAIN maintains high reconstruction quality across different network sizes and depths, and settings.

In summary, our paper is the first to demonstrate that GCN training in a federated learning setting poses data privacy risks. We believe that this is a promising initial step towards identifying these vulnerabilities and developing effective defense mechanisms.

REFERENCES

Mislav Balunovic, Dimitar Dimitrov, Nikola Jovanović, and Martin Vechev. Lamp: Extracting text from gradients with language model priors. *Advances in Neural Information Processing Systems*, 35:7641–7654, 2022.

Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the curious abandon honesty: Federated learning is not private. *arXiv*, 2021.

Hong-Min Chu, Jonas Geiping, Liam H Fowl, Micah Goldblum, and Tom Goldstein. Panning for gold in federated learning: Targeted text extraction under arbitrarily large-scale aggregation. *ICLR*, 2023.

Hejie Cui, Zijie Lu, Pan Li, and Carl Yang. On positional and structural node features for graph neural networks on non-attributed graphs. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 3898–3902, 2022.

Jieren Deng, Yijue Wang, Ji Li, Chao Shang, Hang Liu, Sanguthevar Rajasekaran, and Caiwen Ding. Tag: Gradient attack on transformer-based language models. *arXiv preprint arXiv:2103.06819*, 2021.

Dimitar I Dimitrov, Maximilian Baader, Mark Niklas Müller, and Martin Vechev. Spear: Exact gradient inversion of batches in federated learning. *arXiv preprint arXiv:2403.03945*, 2024.

Scott Fortin. The graph isomorphism problem. 1996.

Liam Fowl, Jonas Geiping, Steven Reich, Yuxin Wen, Wojtek Czaja, Micah Goldblum, and Tom Goldstein. Decepticons: Corrupted transformers breach privacy in federated learning for language models. *ICLR*, 2022a.

Liam H. Fowl, Jonas Geiping, Wojciech Czaja, Micah Goldblum, and Tom Goldstein. Robbing the fed: Directly obtaining private data in federated learning with modified models. In *ICLR*, 2022b.

András Frank. On kuhn's hungarian method—a tribute from hungary. *Naval Research Logistics (NRL)*, 52(1):2–5, 2005.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in neural information processing systems*, 33:16937–16947, 2020.

Jiahui Geng, Yongli Mou, Feifei Li, Qing Li, Oya Beyan, Stefan Decker, and Chunming Rong. Towards general deep leakage in federated learning. *arXiv*, 2021.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

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

Harlin Lee, Andrea L Bertozzi, Jelena Kovačević, and Yuejie Chi. Privacy-preserving federated multi-task linear regression: A one-shot linear mixing approach inspired by graph regularization. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5947–5951. IEEE, 2022.

Zhuohang Li, Jiaxin Zhang, Luyang Liu, and Jian Liu. Auditing privacy defenses in federated learning via generative gradient leakage. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10132–10142, 2022.

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

Liang Peng, Nan Wang, Nicha Dvornek, Xiaofeng Zhu, and Xiaoxiao Li. Fedni: Federated graph learning with network inpainting for population-based disease prediction. *IEEE Transactions on Medical Imaging*, 42(7):2032–2043, 2022.

Ivo Petrov, Dimitar I. Dimitrov, Maximilian Baader, Mark Niklas Müller, and Martin T. Vechev. DAGER: exact gradient inversion for large language models. *CoRR*, abs/2405.15586, 2024. doi: 10.48550/ARXIV.2405.15586. URL https://doi.org/10.48550/arXiv.2405.15586.

Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Trans. Inf. Forensics Secur.*, (5), 2018.

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. *Advances in neural information processing systems*, 33:12559–12571, 2020.

Mark Vero, Mislav Balunović, Dimitar Iliev Dimitrov, and Martin Vechev. Tableak: Tabular data leakage in federated learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 35051–35083. PMLR, 2023.

Yuxin Wen, Jonas Geiping, Liam Fowl, Micah Goldblum, and Tom Goldstein. Fishing for user data in large-batch federated learning via gradient magnification. In *ICML*, 2022.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

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

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M. Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *CVPR*, 2021.

Chenhan Zhang, Shuyu Zhang, JQ James, and Shui Yu. Fastgnn: A topological information protected federated learning approach for traffic speed forecasting. *IEEE Transactions on Industrial Informatics*, 17(12):8464–8474, 2021.

Chi Zhang, Zhang Xiaoman, Ekanut Sotthiwat, Yanyu Xu, Ping Liu, Liangli Zhen, and Yong Liu. Generative gradient inversion via over-parameterized networks in federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5126–5135, 2023.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

Wei Zhu, Jiebo Luo, and Andrew D White. Federated learning of molecular properties with graph neural networks in a heterogeneous setting. *Patterns*, 3(6), 2022.

A ADDITIONAL TECHNICAL DETAILS

A.1 TABLE OF NOTATIONS

For convenience, we add a table of notations containing brief definitons for all symbols used in our work.

Table 5: Table of notations used in the technical description of GRAIN.

Symbol	Definition	Symbol	Definition
$\mathcal{G} = (V, E)$	Graph with nodes V and edges E	n	# of nodes in the graph
$\boldsymbol{A}$	The adjacency matrix	$\tilde{\boldsymbol{A}}$	The normalized adj. matrix
$\text{dist}(v_s, v_e)$	# edges in shortest path connecting nodes $v_s, v_e \in V$	$\mathcal{N}_{\mathcal{G}}^{k}(v)$	Degree-k neighborhood in graph $\mathcal{G}$ with center node v
$\deg_{\mathcal{G}}(v)$	Degree of node v in graph $\mathcal{G}$	$\deg(v)$	Degree of node v as given by its feature
$\boldsymbol{X}^i$	Input to the ith GNN layer	$\boldsymbol{X}_v^i$	i-th layer input feature of node v
$\mathcal{L}$	Loss	$\boldsymbol{W}^i$	Weights of the i-th layer
d'	Hidden dimension size	L	Number of GCN layers
f_i	Function mapping the input graph to the output of the i-th layer	f	# features
$\mathcal{F}_i$	Set of values for the i-th feature	$\mathcal{F}$	$\mathcal{F}_1 \times \cdots \times \mathcal{F}_f$ - set of all possible feature combinations
τ	Span check distance threshold	$\boldsymbol{D}$	Degree matrix (diagonal)
$\mathcal{T}_l$	Proposal set of degree-l building blocks	$\mathcal{T}_l^*$	Filtered set of degree-l building blocks
$\mathcal{T}_B^*$	Final set of filtered building blocks	σ	Activation function
$\Delta_{\mathcal{G}}$	Distance between the gradients of $\mathcal{G}$ and observed gradients	d_{best}	Gradient distance of the best reconstructed graph
GRAPH-N$(\mathcal{G}, \hat{\mathcal{G}})$	Similarity between degree-N neighborhoods of $\mathcal{G}$ and $\hat{\mathcal{G}}$	$\mathcal{G}_{best}$	The best reconstructed graph.

A.2 DEFERRED PROOFS

Here we show the proof of Lemma 5.1, which we restate here for convenience:

Lemma 5.1. *For $\frac{\partial \mathcal{L}}{\partial \boldsymbol{Z}^i}$ of full-rank, $d < n$, and a possibly normalized adjacency matrix at layer i, $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, $\boldsymbol{X}_j^i \in \text{colspan}(\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^i})$ if and only if $\boldsymbol{A}_j^T \notin \text{colspan}(\bar{\boldsymbol{A}}_j)$, where $\bar{\boldsymbol{A}}_j$ denotes the matrix $\boldsymbol{A}$ with its j-th column removed.*

Proof. We separate the proof in 3 steps:

- Step 1: $A_i^T \notin \text{colspan}(\bar{A}_i)$ is equivalent to $\text{null}(\bar{A_i^T}) \not\subseteq \text{null}(A_i^T)$
- Step 2: There is a vector $\boldsymbol{x}_i$, such that $\boldsymbol{x}_i A = e_i$ if and only if $\text{null}(\bar{A_i^T}) \not\subseteq \text{null}(A_i^T)$.
- Step 3: $\boldsymbol{X}_i \in \text{colspan}(\frac{\partial \mathcal{L}}{\partial W})$ if and only if there is a vector $\boldsymbol{x}_i$, such that $\boldsymbol{x}_i^T A = e_i$, where e_i is the i-th standard basis vector.

Step 1: ($A_i^T \notin \text{colspan}(\bar{A}_i) \iff \text{null}(\bar{A_i^T}) \not\subseteq \text{null}(A_i^T)$) First of all, the statement is equivalent to negating both sides, or $A_i^T \in \text{colspan}(\bar{A}_i) \iff \text{null}(\bar{A_i^T}) \subseteq \text{null}(A_i^T)$, which can be shown by the following steps:

$$\begin{aligned} \text{null}(\bar{A_i^T}) \subseteq \text{null}(A_i^T) &\iff \text{null}(\bar{A_i^T}) \subseteq \text{null}(A_i^T) \\ &\iff \text{rowspan}(A_i^T) \subseteq \text{rowspan}(\bar{A_i^T}) \\ &\iff \text{colspan}(A_i) \subseteq \text{colspan}(\bar{A}_i) \end{aligned}$$

$$\iff A_i \in \text{colspan}(\bar{A}_i)$$

Here we used that the complenetary subspace of the null space of matrix is the rowspan of the matrix $\text{null}(M)^C = \text{rowspan}(M)$. The last step follows from that the fact that A_i is a single common vector, and therefore all vectors in $\text{colspan}(A_i)$ are of the form λA_i.

Step 2 ($\text{null}(\bar{A_i^T}) \not\subseteq \text{null}(A_i^T) \iff \exists \boldsymbol{x}_i . \boldsymbol{x}_i^T A = e_i$): First, for both directions of the proof, we can separate $\boldsymbol{x}_i A = e_i$ into 2 different requirements:

$$\boldsymbol{x}_i^T \bar{A}_i = \mathbf{0} \tag{2}$$

$$\boldsymbol{x}_i^T A_i = 1 \tag{3}$$

($\Rightarrow$) First of all, we note that $A_i^T \in \mathbb{R}^{1\times n}$ has $\text{rank}(A_i^T) = 1$, as for a GCN A contains self-loops, meaning that A_i contains a non-zero entry. Therefore, A_i^T has $\text{nullity}(A_i^T) = n - 1$ due to the rank-nullity theorem. $\text{null}(\bar{A_i^T}) \not\subseteq \text{null}(A_i^T)$ implies that there exists an $\boldsymbol{x}_i^{'} \in \text{null}(\bar{A_i^T})$, such that $\boldsymbol{x}_i^{'} \notin \text{null}(A_i^T)$ (since $\text{nullity}(A_i^T) = n - 1 < n$ this set is non-empty). For that $\boldsymbol{x}_i$, the following hold:

$$\boldsymbol{x}_i^{'T} \bar{A}_i = \mathbf{0}$$
$$\boldsymbol{x}_i^{'T} A_i = c$$

Therefore, if we take $\boldsymbol{x}_i = \frac{1}{c}\boldsymbol{x}_i^{'}$, $\boldsymbol{x}_i$ would satisfy both (1) and (2), giving us a valid solution.

($\Leftarrow$) Assuming the existence of $\boldsymbol{x}_i$ with $\boldsymbol{x}_i^T A = e_i$, we know that (1) and (2) hold. Equivalently to (1), $\boldsymbol{x}_i \in \text{null}(\bar{A_i^T})$. If we assume the converse of $\text{null}(\bar{A_i^T}) \not\subseteq \text{null}(A_i^T)$, which is $\text{null}(\bar{A_i^T}) \subseteq \text{null}(A_i^T)$, then $\boldsymbol{x}_i$ is also in $\text{null}(A_i^T)$. This would imply that $\boldsymbol{x}_i^T A_i = 0$, which contradicts (2). Therefore, by contradiction $\text{null}(\bar{A_i^T}) \not\subseteq \text{null}(A_i^T)$ holds, concluding the proof of this step.

Step 3 ($\exists \boldsymbol{x}_i . \boldsymbol{x}_i^T A = e_i \iff \boldsymbol{X}_i \in \text{colspan}(\frac{\partial \mathcal{L}}{\partial W})$): ($\Rightarrow$) Assuming the existence of such an $\boldsymbol{x}_i$ implies that we can multiply both sides of the equation by $\boldsymbol{X}$ to obtain $\boldsymbol{x}_i^T A \boldsymbol{X} = \boldsymbol{X}_i$. This implies that $\boldsymbol{X}_i \in \text{rowspan}(A\boldsymbol{X})$. Applying Theorem DAGER(TODO: change) on $\frac{\partial \mathcal{L}}{\partial W} = (AX)\frac{\partial \mathcal{L}}{\partial Z}$, implies that $\text{rowspan}(AX) = \text{colspan}(\frac{\partial \mathcal{L}}{\partial W})$, and therefore $\boldsymbol{X}_i \in \text{colspan}(\frac{\partial \mathcal{L}}{\partial W})$.

($\Leftarrow$) Applying Theorem DAGER(TODO: change) on $\frac{\partial \mathcal{L}}{\partial W} = (AX)\frac{\partial \mathcal{L}}{\partial Z}$, or $\text{rowspan}(AX) = \text{colspan}(\frac{\partial \mathcal{L}}{\partial W})$, implying that $\boldsymbol{X}_i \in \text{rowspan}(A\boldsymbol{X})$. This can be rewritten as $\exists \boldsymbol{x}_i . \boldsymbol{x}_i^T A\boldsymbol{X} = \boldsymbol{X}_i$. Assuming $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ is full-rank, then there exists a right-inverse $\boldsymbol{X}^{-R}$, as $\text{rank}(\boldsymbol{X}) = d < n$.

$$\boldsymbol{x}_i^T A \boldsymbol{X} \boldsymbol{X}^{-R} = \boldsymbol{X}_i \boldsymbol{X}^{-R} \Rightarrow \boldsymbol{x}_i A = e_i$$

It is notable that $\boldsymbol{X}$ not being full-rank still allows for all nodes with feature vectors in $\boldsymbol{X}$ will pass the span check, however it is possible that some hallucinated inputs might also pass the check. This concludes our proof.

□

A.3 Depth-First Search implementation

B Additional Experiments

Here we present additional experiments that are not part of the main text.

B.1 Human evaluation for the GRAPH set of metrics

We performed a human evaluation, where 3 experts in Graph Theory and Chemistry were shown 120 sample reconstructions of molecules, as given by DLG and GRAIN. The samples were shuffled, and the participants were tasked to assign a score from 0 to 10, with the following instructions:

Algorithm 5 Depth-first search reconstruction

function DoDFS($\mathcal{T}_B^*, \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}}, \mathcal{G}_0, Y$)
  $\mathcal{G}_{\text{top}} \leftarrow \emptyset$
  $d_{top} \leftarrow \infty$
  $S^*_{\text{new}} \leftarrow \{\}$

  if $|dang(\mathcal{G}_0)| == 0$ **then**
    $d_0 \leftarrow \min_{y \in Y} \| \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} - \frac{\partial \mathcal{L}(\mathcal{G}_0, y)}{\partial W} \|_{\text{F}}$
    return $d_0, \mathcal{G}_0$

  $v = dang(G_0)[0]$
  for $\mathcal{G}_1$ in $\mathcal{T}_B^*$ **do**
    if $\exists \mathcal{G}_2 = \text{glue}(\mathcal{G}_0, \mathcal{G}_1, v)$ **then**
      $S^*_{\text{new}} \leftarrow S^*_{\text{new}} \cup \{\mathcal{G}_2\}$
      for $S \subseteq \{V(\mathcal{G}_2) \setminus V(\mathcal{G}_0)\} \times V(\mathcal{G}_0)$ **do**
        $S^*_{\text{new}} \leftarrow S^*_{new} \cup \{\text{overlap}(\mathcal{G}_2, S)\}$

  for G in S^*_{new} **do**
    $d', \mathcal{G}' \leftarrow$DoDFS($\mathcal{T}_B^*, \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}}, \mathcal{G}$)
    if $d' == 0$ **then**
      return $0, \mathcal{G}'$
    else if $d_{\text{TOP}} > d'$ **then**
      $d_{\text{TOP}}, \mathcal{G}_{\text{TOP}} \leftarrow d', \mathcal{G}'$

  return $d_{\text{TOP}}, \mathcal{G}_{\text{TOP}}$

"Thank you for agreeing to participate in this study on the quality of graph reconstructions! We have gathered a set of graphs, coupled with the best-effort reconstruction. Please give each pair a score of 0-10, where 0 is a complete lack of similarity, and 10 is a perfect match. When assigning a score, take into account the *structure* of the two graphs, as well as the *atom type* for matching atoms, and also be wary that 2 graphs might be *isomorphic*, but have different pictures. Please disregard the connections between atoms, as the methods we used do not recover any edge properties. Give your, as best as possible, score on how similar the graphs are with respect to these properties."

We report the average scores for each algorithm, multiplied by a factor of 10 to match the order of magnitude of the GRAPH metrics, and present the results in Table 6.

Table 6: Comparison of the designed metrics with the human evaluation.

	GRAPH-0	**GRAPH-1**	**GRAPH-2**	**Study score**
Ours	**72.6**	**67.8**	**66.9**	**70.6**
DLG	24.2	10.5	12.0	6.5

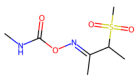
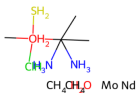
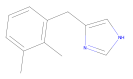
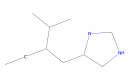
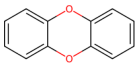
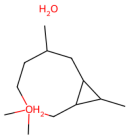
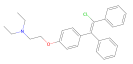
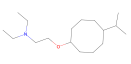
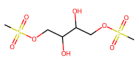
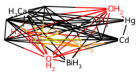
Based on these studies, we also show in Table 7 that our partial reconstructions are deemed more significant than what the metric suggests, likely meaning that there are examples which present significant information leakage. In contrast, high-scoring examples from the DLG attacks have been rated as essentially uninformative.

B.2 Ablation Studies

We perform additional ablation studies on various assumptions and parameters.

First, we investigate the effect of the choice for the τ threshold, used for filtering inputs using the span check method. We measure the ratio between the number of nodes and degree-1 building blocks that pass the filter, and the actual number of these blocks. That is done on 10 samples from the Tox21 dataset, for $\tau \in [10^{-6}, 0.1]$. We show in Fig. 4 that any $\tau \in [10^{-4}, 10^{-2}]$ results in

Table 7: Score discrepancy examples between human evaluators and the GRAPH set of metrics. G-1 stands for the GRAPH-1 metric.

GT	GRAIN	G-1	Study	GT	DLG	G-1	Study
		62.0	93.3			52.7	10.0
		41.3	63.3			61.0	23.3
		33.5	56.7			41.0	0.0

essentially the same filtering process, and that thresholds in this interval perfectly recover the correct degree-1 building blocks.

Additionally, we note that GRAIN is not significantly impacted by the embedding dimension d', as long as $n < d'$, consequently achieving similar scores, particularly for small graphs. We show the exact results in Table 8.

Table 8: Results (in %) of GRAIN with different embedding dimensions across a range of graph sizes

	$n \leq 15$			$16 \leq n \leq 25$			$26 \leq n$		
	GRAPH-0	GRAPH-2	FULL	GRAPH-0	GRAPH-2	FULL	GRAPH-0	GRAPH-2	FULL
$d = 300$	$\mathbf{93.0}^{+3.4}_{-5.4}$	$\mathbf{91.6}^{+3.8}_{-6.3}$	$\mathbf{81.9 \pm 1.7}$	$\mathbf{81.7}^{+3.9}_{-4.8}$	$\mathbf{74.8}^{+5.8}_{-6.3}$	$\mathbf{43.6 \pm 1.1}$	$50.1^{+6.8}_{-7.1}$	$39.2^{+8.5}_{-7.7}$	$\mathbf{5.1 \pm 0.6}$
$d = 128$	$\mathbf{92.1}^{+3.2}_{-5.0}$	$\mathbf{92.3}^{+3.9}_{-5.7}$	$\mathbf{79.3 \pm 1.6}$	$\mathbf{81.4}^{+4.0}_{-4.8}$	$\mathbf{75.1}^{+5.7}_{-6.6}$	$\mathbf{43.6 \pm 1.1}$	$49.3^{+7.2}_{-6.5}$	$38.8^{+8.7}_{-7.6}$	$\mathbf{5.1 \pm 0.6}$
$d = 64$	$\mathbf{92.2}^{+3.0}_{-5.5}$	$\mathbf{92.0}^{+4.0}_{-5.9}$	$\mathbf{79.3 \pm 1.6}$	$\mathbf{81.3}^{+4.1}_{-4.7}$	$\mathbf{75.5}^{+5.8}_{-6.5}$	$\mathbf{43.6 \pm 1.1}$	$48.6^{+7.4}_{-6.5}$	$37.9^{+9.0}_{-7.7}$	$\mathbf{5.1 \pm 0.6}$
$d = 32$	$\mathbf{92.2}^{+3.0}_{-5.5}$	$\mathbf{91.7}^{+3.6}_{-6.5}$	$\mathbf{79.3 \pm 1.6}$	$\mathbf{81.7}^{+4.0}_{-4.4}$	$\mathbf{73.8 \pm 6.1}$	$\mathbf{43.6 \pm 1.1}$	$15.3^{+2.8}_{-4.4}$	$13.3^{+2.5}_{-3.9}$	$\mathbf{0.0 \pm 0.0}$

Further, in Fig. 5 we investigated how the rank-deficiency of the adjacency matrix A affects the strength of the GRAIN adversary. For different sizes of A, we measure what the Monte-Carlo probability of A being full-rank, and the fraction of nodes we can recover, as computed per Lemma 5.1. This was done for synthetic graphs, where we sampled 100,000 symmetric binary matrices with varying probability of every 2 nodes being connected, as well as for all molecular graphs in the chemical datasets Clintox, Tox21 and BBBP. We show that Lemma 5.1 is crucial for understanding why GRAIN is effective, despite the probability of A being full-rank being low. In particular, we highlight in Fig. 5 that GRAIN can recover an increasing fraction of nodes as A grows.

Finally, we compare the computational cost of GRAIN to that of the baseline attacks. We ensured that the optimization attacks reached convergence before terminating each sample search. We observe in Table 9 that GRAIN achieves significantly better results (seen on Table 1), despite running for time comparable to the one of Tableak.

B.3 Additional settings

Here we present our results on different applications of GRAIN under the Graph Attention Network (GAT) architecture on both the Tox21 chemical dataset, and the Citeseer dataset. In Table 10 we show these additional experiments, alongside the effects of running GRAIN without knowledge of the node degree, or without utilisation of the node uniqueness heuristic.

Figure 4: Ablation study on the span check filtering threshold τ.

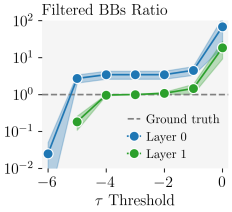


Figure 5: Impact of low-rankness of the adjacency matrix on reconstructability for synthetic data (left) and molecules (right)

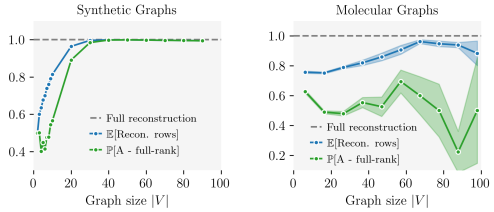


Table 9: Runtime for each GRAIN and baseline experiment, given in hours.

	GRAIN	DLG	DLG+A	Tableak	Tableak+A
Tox21	14.3	3.3	3.1	13.1	12.3
Clintox	24.1	3.5	3.2	15.2	14.5
BBBP	23.7	3.9	3.1	12.6	12.5

Table 10: Results (in %) of GRAIN in the following cases: Row 1 – the original setting, Row 2 – the activation function GELU instead of ReLU, Row 3 – in the gradients shared are from a trained model instead of the first epoch, Row 4 – results of GRAIN in node classification task

	GRAPH-0	GRAPH-1	GRAPH-2	FULL
GAT, Tox21	$92.9^{+3.8}_{-5.8}$	$90.7^{+5.0}_{-7.1}$	$89.9^{+5.8}_{-7.2}$	75.0 ± 1.8
GAT, Citeseer	$79.3^{+4.7}_{-6.3}$	$69.1^{+6.1}_{-6.4}$	$69.6^{+6.2}_{-6.0}$	61.0 ± 1.6
GAT, Citeseer,no degree	$59.7^{+6.8}_{-7.2}$	$42.7^{+6.3}_{-6.6}$	$43.2^{+6.4}_{-6.6}$	32.0 ± 1.1
GAT, Citeseer,no heuristic	$64.6^{+3.5}_{-4.2}$	$52.1^{+4.7}_{-5.3}$	$52.4^{+4.6}_{-5.2}$	44.0 ± 1.3