Investigating Differentially Private Graph Neural Networks with Weighted Aggregations

Frederick Shpilevskiy
University of British Columbia
fshpil@students.cs.ubc.ca

Andre Mello
University of British Columbia
amellow@student.ubc.ca

Michael Liu
University of British Columbia
mfliu@cs.ubc.ca

Abstract

In the realm of graph-structured data, Graph Neural Networks (GNNs) have gathered significant attention by outperforming standard machine learning techniques. However, GNNs have been shown to be vulnerable to privacy attacks. There has been extensive work building GNNs that satisfy Differential Privacy (DP) guarantees. GNN based on Aggregation Perturbation (GAP) is a proposed approach by Sajadmanesh et al. that injects noise into the aggregation function of a GNN. We propose and investigate learned and heuristic weighted aggregation methods. We present the privacy cost of learned methods and their resulting poor performance. We propose heuristic similarity-based weightings that require no extra privacy costs and show promising performance and opportunity for future investigation.

1 Introduction

Graph Neural Networks (GNNs) are a powerful form of neural network that operates on graph structured data. At a high level, given a graph with node features and edges, a GNN produces new embeddings for each node. These embeddings encode information about the node itself, along with the graph structure around it. This approach makes GNNs useful in handling graph-specific tasks such as clustering, link prediction, and node and edge classification. GNNs have become a widely used method due to their superior performance and flexibility in such tasks. They have found application in analysis of social networks, biology, and molecular chemistry. Recently, they are also seeing use in traffic prediction [1], fake news detection [2], and recommendation systems [3].

Many applications of GNNs exist in fields such as medicine and social network analysis where datasets contain sensitive information about individuals and their relations with others. However, GNNs have been shown to be vulnerable to privacy attacks such as membership inference attacks [4]. As a result of encoding structural information in node embeddings, GNNs are especially vulnerable to these kinds of attacks [5]. Strong privacy guarantees are therefore necessary to ensure the safety of training data.

Differential Privacy (DP) [6] is a rigorous definition of privacy that ensures the released information of an algorithm is guaranteed to uphold the privacy of any individual's data in the input. In approximate DP [7], the output of a differentially private algorithm has a high probability to not reveal the membership status of any individual in the input dataset significantly. These definitions of privacy are additionally robust against attacks which use auxiliary knowledge, such as related data and priors. Furthermore, DP guarantees are maintained through post-processing, even if the post-processing steps are not differentially private themselves. This means that no further algorithm or analysis can

Preprint. Under review.

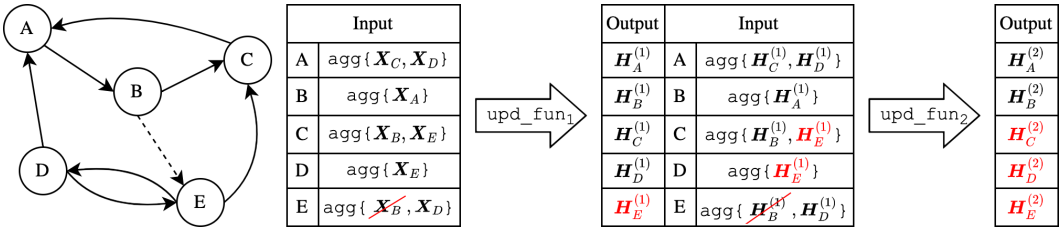

Figure 1: A diagram of a 2-layer GNN with the left graph as an input. The input to each layer is an aggregation of a single node's neighbours' embeddings. For the first layer, we use node features instead. The output is the embedding of that node. The diagram displays edge-level sensitivity through how change propagates throughout the network when an edge (B, E) is removed. All affected embeddings are shown in red. Adapted from [13]

$$\boldsymbol{H}_v^{(k)} = \texttt{upd\_fun}_k\{\texttt{agg}\{\boldsymbol{H}_u^{(k-1)} : \forall u \in N_v\}\} \tag{1}$$

Equation 1: An equation demonstrating the message passing algorithm. $\boldsymbol{H}_v^{(k)}$ is the output embedding of node v for layer k. N_v is the set of nodes u that have an edge from u to v. $\texttt{agg}\{\cdot\}$ is some aggregation function, such as $\texttt{Mean}$ or $\texttt{Sum}$, that takes a list of node embeddings as input. $\texttt{upd\_fun}_k\{\cdot\}$ is some learnable update function, such as an MLP, for layer k that takes the aggregation of a node's neighbours' embeddings as an input and outputs the node's new embedding.

extract more information about the individuals in the dataset. Lastly, differentially private algorithms can be composed to produce new algorithms whose privacy guarantees can be bounded and analyzed.

While the composition strategies introduced in [7], [8], and [9] are able to provide some differential privacy guarantees, a different formulation, namely Renyi Differential Privacy (RDP) [10] is able to show stronger results in the context of deep learning. To analyze the composition of many steps of an algorithm, we use the analytical moments accountant [11], a utility based on RDP to keep track of spent privacy budget through the course of training. An implementation of this accountant is provided by the PyVacy [12] library which we use in our experiments.

GNNs pose a unique challenge to DP as a result of the message passing algorithm. Message passing works by first gathering a given node's neighbours' embeddings. In Equation 1, this is shown in red. Then, we aggregate them via some function into a pooled "message". This is shown in blue. Finally, we pass that message into an update function to generate a new embedding for the chosen node [14]. This is shown in purple in Equation 1. If this algorithm is executed K times, the embedding will store structural information of the chosen node's K-hop neighbourhood. As a result, a node embedding becomes sensitive to all changes pertaining to neighbours within that node's K-hop neighbourhood. Figure 1 demonstrates how removing edge (B, E) causes changes that propagate with increasing hops. This propagating change makes the task of bounding the sensitivity of a function that depends on the embeddings a non-trivial matter. This effect also prevents nodes from being sampled independently, which violates the sampling requirements of DP algorithms such as DP-SGD.

There exist many approaches for guaranteeing DP with graph-structured data. Two common definitions of privacy in these contexts are edge-level and node-level differential privacy. Edge-level privacy guarantees DP across neighbouring graph-structured datasets that differ by one edge while node-level privacy differs by one node and all edges associated with it. In this sense, node-level privacy is a strictly stronger DP guarantee. Some approaches to DP-GNN include perturbing the adjacency matrix of an input graph [15], employing knowledge distillation [16], and adapting DP-SGD to GNNs [17]. A novel approach to DP-GNNs is GNN based on aggregation perturbation (GAP) proposed by Sajadmanesh et al. [13]. GAP is able to guarantee edge-level privacy and node-level privacy at the cost of some utility. It has demonstrated superior performance to above mentioned approaches (specifically [15], [17]).

Our work builds on Sajadmanesh et al. [13]'s DP-GNN architecture. We investigate weighted aggregation with both learned and heuristic weightings. Our approach focuses on the edge-level privacy regime. Specifically, our work has the following contributions:

- We show through our proof and experiments the drawbacks of learning aggregations weightings privately; in particular, we explain how this approach is incompatible with the original model architecture
- We demonstrate that weightings using heuristic similarity metrics have superior performance to learned methods with no extra privacy costs

2 Background

2.1 Graph Differential Privacy

Differential privacy is a formal notion of the amount of information leaked by an algorithm about any individual from an input dataset. It considers two datasets differing by a single entry and bounds the distance between the output distributions of the algorithm on these datasets.

Definition 1 (Dwork et al. [7])**.** *A randomized algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private if for all pairs of adjacent datasets D and D', and for any possible output $o \in Range(\mathcal{A})$, we have*

$$\Pr(\mathcal{A}(D) = o) \leq e^{\epsilon} \Pr(\mathcal{A}(D') = o) + \delta$$

The following are two common definitions of adjacent graphs:

1. Two graphs G and G' are *edge-level adjacent* if one can be obtained from the other by removing one edge.
2. Two graphs G and G' are *node-level adjacent* if one can be obtained from the other by removing one node, including its features, incident edges, and label.

2.2 GNN based on Aggregation Perturbation (GAP)

GAP is a DP-GNN architecture proposed by Sajadmanesh et al. [13]. GAP ensures edge-level privacy by perturbing the output of the aggregation function – effectively hiding the influence of neighbouring nodes. It can be extended to node-level privacy by bounding the degree of each node to limit the sensitivity of the aggregation function, then using a DP learning algorithm such as DP-SGD [11] during training. The structure of GAP is summarised with 3 main components: an MLP model as an encoder that extracts low-dimensional node features while ignoring graph structure [13, Lines 1-2 in Algorithm 2], an aggregation module that perturbs the output of the aggregation of the encoded features and K-hop neighbourhood [13, Lines 3-4 in Algorithm 2], and a classification module that uses aggregated features for classification tasks instead of querying graph edges [13, Line 5 in Algorithm 2].

The encoder module is pre-trained alone for label prediction to significantly reduce the privacy cost in the aggregation module. The aggregation module applies a perturbation using the Gaussian mechanism after each aggregation step to prevent any interdependency between nodes in the node feature vectors. It also applies a normalization to bound the effect of any single feature vector of a node on subsequent aggregations. The classification module trains a series of multi-layer perceptrons (MLP) using each set of node feature matrices to generate the node embeddings for that respective k-hop where $k \in 0, 1, \ldots, K$. It then aggregates them to train a "head" MLP to predict node labels using posterior probabilities. As a result, the embedding that ignores structure (when $k = 0$) and all other k-hop embeddings are generated using independent MLPs. This effect ensures that regardless of the scale of noise in the aggregation module, the model will always be able to exploit all node-specific, structure-independent information.

GAP was evaluated in the transductive setting in the edge-level privacy regime against LapGraph proposed by Wu et al. [15] and against DP-GCN proposed by Daigavane et al. [17] in the node-level privacy regime. GAP's performance is superior in both node-level and edge-level privacy regimes. GAP provides flexibility in choice of DP guarantees, providing both edge-level and node-level privacy.

2.3 Graph Attention Networks

$$\boldsymbol{H}_v^{(k)} = \sigma \sum_{\forall u \in N_v} \alpha_{u,v}^{(k)} \boldsymbol{W}^{(k)} \boldsymbol{H}_u^{(k-1)} \tag{2}$$

$$\alpha_{u,v}^{(k)} = \texttt{softmax}_{\texttt{u}}\{a(\boldsymbol{W}^{(k)} \boldsymbol{H}_u^{(k-1)}, \boldsymbol{W}^{(k)} \boldsymbol{H}_v^{(k-1)}; \boldsymbol{\Theta}_{\text{att}})\} \tag{3}$$

Equation 2: An equation demonstrating the attention mechanism. $\boldsymbol{H}_v^{(k)}$ is the output embedding of node v for attention layer k. N_v is the set of nodes u that have an edge from u to v. σ is a non-linearity. $\alpha_{u,v}^{(k)}$ is the weight outputted by the $\texttt{softmax}\{\cdot\}$ of a learnable attention function a with parameters $\boldsymbol{\Theta}_{\text{att}}$, at layer k for node v and neighbour u, represented in Equation 3. $\boldsymbol{W}^{(k)}$ are the weights of a learnable linear transformation. $\boldsymbol{H}_u^{(k-1)}$ is a neighbour u's embedding.

Graph Attention Network (GAT), proposed by Veličković et al. [18], is a particular case of the Graph Convolutional Network (GCN) [19] architecture that weighs the embeddings of a node's neighbours. GAT seeks to handle arbitrarily structured graphs by learning an embedding for each node from a learned weighted aggregation of its neighbours' embeddings. The attention layer first transforms the features of a node using a learned weight matrix shared between all nodes. In Equations 2 and 3, the weight matrix for layer k is $\boldsymbol{W}^{(k)}$ and the transformed features of node v at that layer are given by $\boldsymbol{W}^{(k)}\boldsymbol{H}_v^{(k-1)}$. The transformed features are passed through a learned attention mechanism to calculate an attention coefficient for that edge. The attention coefficient in Equation 3 is the output of attention function a given the node's and neighbours' transformed features. Then, the attention coefficients of all edges are passed through the $\texttt{softmax}$ function to get attention weights. Finally, the transformed features of a node's neighbours are weighted by their respective attention weight, summed, and passed through a non-linearity to get the node's embedding. This procedure is shown in Equation 2. As a result of this mechanism, GAT significantly improves upon the performance of other GNN models in the inductive learning setting. Introducing such a design to DP-GNN may improve the utility in the more challenging inductive setting [20], [21].

2.4 Related Work

Link Private Graph Network (LPGNet) is another edge-level DP-GNN proposed by Kolluri et al. [20] that unlike conventional GNNs does not rely on a graph's adjacency matrix. Instead, it assumes data homophily – examples with similar features will be close together in the graph. LPGNet iteratively updates a clustering on the nodes by their predicted label. Each iteration uses an MLP to predict the label of each node based on its previous embedding vector and a degree vector. For each node, this degree vector is calculated by finding, for each label, the number of the node's neighbours that share that label. Since the degree vector depends on the private graph edges, LPGNet injects Laplace distributed noise in this step. However, the sequential composition of many MLPs amplifies the privacy loss; a problem avoided by GAP through pre-training an encoder to produce fixed node embeddings.

GAP has been compared against other DP GNN methods [13]. The GNN model proposed in [22] provides a local node DP guarantee, but cannot provide edge DP due to its reliance on a trusted central server. The issue of edge DP is addressed using randomized response by [23], but it does not provide node DP. A private aggregation of teacher ensembles method is used in [16] to provide node DP, but requires a public graph whose data is not protected. Each of these shortcomings is addressed by GAP, which can provide edge and node DP while not requiring a public graph in training.

3 Our Approach

This section describes our two main approaches to aggregation weighting. The first approach described is our learned attention-based weighting. We explain the drawbacks of this approach and suggest an improved approach that uses similarity metrics.

3.1 Learned Weighting

Our first approach is an attention-based weighted aggregation. Our goal in this approach is to adapt the attention mechanism used in GATs [18] to the private setting. Given that the attention is learned, for this approach, we focus on edge-level privacy. We implement the attention mechanism within GAP's aggregation module in order to keep the classification and encoder modules safe from accessing sensitive edge information.

Algorithm 1 Attention-Weighted Aggregation

Input: A set of nodes U (the red nodes in Figure 2), A set of incoming edges EL (the blue edges of 2), A set of neighbour nodes N (the blue nodes of 2), Node embeddings $\boldsymbol{X}^{(0)}$ outputted by encoder for nodes $U \cup N$, Noise scale σ^2, Model parameters $\boldsymbol{\Theta}_{\text{att}}$ of the attention model

Output: Set of node embeddings $\{\check{\boldsymbol{X}}^{(0)}, \check{\boldsymbol{X}}^{(1)}\}$

1: **for** $v \in U \cup N$ **do**
2: $\quad \check{\boldsymbol{X}}_v^{(0)} \leftarrow \boldsymbol{X}_v^{(0)} / \|\boldsymbol{X}_v^{(0)}\|_2$ ▷ Row-normalize the embeddings
3: **end for**
4: **for** $(n, u) \in EL \mid n \in N, u \in U$ **do**
5: $\quad \boldsymbol{e}_{(n,u)} \leftarrow a(\check{\boldsymbol{X}}_n^{(0)}, \check{\boldsymbol{X}}_u^{(0)}; \boldsymbol{\Theta}_{\text{att}})$ ▷ Compute attention coefficients
6: $\quad \boldsymbol{\alpha}_{(n,u)} \leftarrow \texttt{sigmoid}(\boldsymbol{e}_{(n,u)})$ ▷ Compute attention weightings
7: **end for**
8: **for** $u \in U$ **do**
9: $\quad \boldsymbol{X}_u^{(1)} \leftarrow \sum_{\forall n \in N \mid (n,u) \in EL} \boldsymbol{\alpha}_{n,u} \check{\boldsymbol{X}}_n^{(0)}$ ▷ Aggregate
10: $\quad \tilde{\boldsymbol{X}}_u^{(1)} \leftarrow \boldsymbol{X}_u^{(1)} + \mathcal{N}(\sigma^2)$ ▷ Perturb
11: $\quad \check{\boldsymbol{X}}_u^{(1)} \leftarrow \tilde{\boldsymbol{X}}_u^{(1)} / \|\tilde{\boldsymbol{X}}_u^{(1)}\|_2$ ▷ Row-normalize
12: **end for**
13: **return** $\{\check{\boldsymbol{X}}^{(0)}, \check{\boldsymbol{X}}^{(1)}\}$

In order to learn the weights, we train an attention model a on edges. An attention model pass is represented by lines 4 to 7 in Algorithm 1. The input to the attention model is a concatenation of the node features of the source and destination nodes. For a given edge, the model outputs attention coefficients (line 5). The coefficients pass through a sigmoid non-linearity that bounds the attention weighting of the corresponding edges between zero and one (line 6).

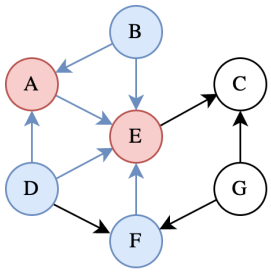

Figure 2: A diagram of a batch selected according to a node selection. The red nodes A, E are selected. For this batch, these are the nodes which we classify. The red nodes' incoming edges and neighbours, in blue, are included in the batch. These are the nodes we use for the aggregation to help predict the class of the red nodes.

We now describe how a node's embedding is computed from it's neighbours' features. Line 9 of Algorithm 1 demonstrates the first step of computing a node v's embedding. We compute a weighted aggregation of v's neighbours' features according to v's incoming edges – we assume that the message is passed from the destination node to the source node. Following the same procedure as GAP's aggregation module, on line 10, we apply Gaussian noise to v's aggregated node features $\boldsymbol{X}_v^{(1)}$. Finally, on line 11, we row-normalize the perturbed aggregations.

We use a sigmoid instead of the softmax used in GAT [18]. There are two reasons for this decision. First, GAP struggles on low-degree nodes because the aggregation is overpowered by Gaussian noise. If all of the weights must add up to one and each node's features are normalized, then the aggregations will be smaller than the unweighted case and will all be overpowered by noise – instead of just the low-degree aggregations. So, to still use weightings but not reduce the impact of all edges on the aggregation, we bound the weight of an edge between 0 and 1 independently. Second, the softmax violates our DP requirements as the sensitivity is larger. This is because the sensitivity corresponding

to the removal of one edge does not just depend on that edge's weighting, but also the change in weighting of all the other edges as the removed edge's weighting is shifted to the others.

In order to train the attention model, we attach a linear layer and softmax to the aggregation module described in Algorithm 1. For a single training step, for some batch of nodes $U \cup N$ and edges EL, the outputs of 1 are passed to the linear layer and softmax to predict the label of the nodes in set U. Using a DP optimizer, we update the parameters of the attention model. We train the attention model over T iterations. At that point, we remove the attached linear layer, do one more pass over the aggregation module with the learned attention model using all of the nodes in the graph, and save the outputs to a cache to be used by the classification module described in 4.4 of [13].

The main limitation of this weighting approach is the privacy cost of training. The full proofs are provided in Appendix A.

Theorem 2. *The edge-level privacy cost of training the attention model is* $(\alpha, \epsilon_1(\alpha) + T\alpha/2\sigma^2)$*-RDP.*

It is clear that this privacy bound is not ideal. In order to train the attention model for T iterations, we must do the aggregation step T times. GAP intended the aggregation step to only be done once. The output of the aggregation module is cached so that the DP mechanism is only composed K times. This represents a significant savings in the privacy budget that our method is unable to utilize.

Another disadvantage of the learned weightings is that training the attention model requires a careful selection of batches. Since the attention model is trained on edges (which are sensitive in edge-level privacy), we must use DP-SGD. We must ensure that batches of edges are independent from one another and all edges have a probability q of being selected in a batch. Furthermore, we must ensure that for a given node, as many of its incoming edges are selected as possible so as to avoid the low-degree aggregation problem. So, we use the batching scheme presented in Figure 2. We first select a fixed number of nodes; in Figure 2, these are the two red nodes. We then select all of their incoming edges and respective neighbour nodes – represented in blue in the figure. This batching method ensures that the probability of a given edge to be in the batch is bounded by the probability of it's destination node to be selected. However, this batching technique does not work for multi-hop aggregations. Therefore, we focus only on 1-hop aggregations for this weighting method.

In the next section, we introduce an alternative weighting scheme that has a significantly better privacy cost.

3.2 Similarity-based Weighting

The previous approach is disadvantageous as a result of the privacy cost brought about by model training. So, instead of learning the weights with an attention model, we develop a heuristic weighting scheme. We assume that the importance of an edge to the aggregation is based on the similarity of the edge's source node to the aggregated node. This assumption is motivated by the idea behind GNNs – we assume that nodes are similar to their neighbours. We believe that by reducing the contribution of non-similar nodes to the aggregation, we will make the aggregations more meaningful.

Algorithm 2 Similarity-Weighted Aggregation (Cosine Similarity)

Input: A set of nodes V, A set of incoming edges E, Node embeddings $\boldsymbol{X}^{(0)}$ outputted by encoder, Noise scale σ^2, K hops

Output: Set of node embeddings $\{\check{\boldsymbol{X}}^{(0)}, \ldots, \check{\boldsymbol{X}}^{(K)}\}$

for $v \in V$ **do**
  $\check{\boldsymbol{X}}_v^{(0)} \leftarrow \boldsymbol{X}_v^{(0)} / \|\boldsymbol{X}_v^{(0)}\|_2$ ▷ Row-normalize the embeddings
end for
for $k \in [0, K)$ **do**
  for $v \in V$ **do**
    $\boldsymbol{X}_v^{(k+1)} \leftarrow \sum_{\forall u \in V \mid (u,v) \in E} \texttt{sigmoid}(\check{\boldsymbol{X}}_u \check{\boldsymbol{X}}_v^T)\, \check{\boldsymbol{X}}_u^{(k)}$ ▷ Aggregate
    $\tilde{\boldsymbol{X}}_v^{(k+1)} \leftarrow \boldsymbol{X}_u^{(k)} + \mathcal{N}(\sigma^2)$ ▷ Perturb
    $\check{\boldsymbol{X}}_v^{(1)} \leftarrow \tilde{\boldsymbol{X}}_u^{(k)} / \|\tilde{\boldsymbol{X}}_u^{(k)}\|_2$ ▷ Row-normalize
  end for
end for
return $\{\check{\boldsymbol{X}}^{(0)}, \ldots, \check{\boldsymbol{X}}^{(K)}\}$

We use a weighting based on the similarity of the node and it's neighbours. We experiment with the cosine similarity metric. This similarity metric is an excellent fit since the node features are already normalized and often high-dimensional.

Algorithm 2 demonstrates the aggregation module weighted by cosine similarity. For similar reasons to the first approach, we use a sigmoid function to bound the weights between 0 and 1 independently. As well, unlike the learned weighting, we are able to do multi-hop (K-hop) aggregations. On line 2, we row-normalize the encoded node features. For each hop, we aggregate the cosine-similarity-weighted node embeddings of the neighbours according to the equation on line 6. On line 7, we perturb the result using the Gaussian mechanism. Finally, on line 8, we row-normalize the result to get the node embedding for that hop.

Similarly to the other aggregations, we pass the encoded features through Algorithm 2 to generate the cache. We then train the classification module on the cache, avoiding incurring any additional edge-level privacy costs.

The main improvement of this approach over the last is the privacy cost.

Theorem 1. *Assuming similarity metrics bounded between 0 and 1, one pass over the similarity-weighted aggregation has an edge-level privacy cost of* $(\alpha, K\alpha/2\sigma^2)$*-RDP.*

It is clear that since T is usually greater than K, the privacy cost of using a heuristic-based weighting is lower. In the next section, we compare the two approaches experimentally.

4 Experiments

All experiments were conducted using PyTorch and PyTorch Geometric running on an NVIDIA A4000 GPU. We use PyVacy's[12] private optimizers and privacy accountant.

4.1 Datasets

Our methods were evaluated on the following public datasets:

1. **Reddit [24].** In this dataset, we infer which Reddit community a post belongs to. The features are an embedding of the post's text, and the edges are present if there is a user that commented on both posts. In addition, we filtered the dataset to only include classes with more than 1000 examples so as to be consistent with the GAP paper.
2. **Amazon Computers [25].** In this dataset, we map a product to its Amazon categories. The node features are a bag of words representation of a product's reviews; edges represent products that are frequently bought together.
3. **FacebookPagePage [26].** In this dataset, we classify a Facebook page. Node features are a website's bag of words representation; edges represent mutual likes between the pages.

4.2 Preprocessing

The Reddit dataset was filtered to include only classes with more than 1000 examples to be consistent with the GAP paper. All datasets were split into a training set containing 80% of the nodes and a test set containing 20% of the nodes. Edges between nodes in the two sets were removed. The adjacency matrix of each split only contained an edge if the nodes that are connected by that edge were in the same split.

4.3 Model Hyperparameters

To compare all models fairly, since the attention-weighted aggregation does only permit 1-hop aggregations, we only consider 1-hop aggregations. For our tests, the encoder approximately halved the number of features of the input twice using a two-layered MLP. The attention mechanism was a two-layered MLP that halved the size of the inputs before returning a scalar. The DP optimizer we used to train the attention model was DP-Adam. We fixed the number of private training iterations to 1000 and the DP optimizer's noise scale to $\sigma = 1.3$. From these settings, we calculated the ϵ training budget. The remaining privacy budget was used for Gaussian aggregation perturbation. For the other

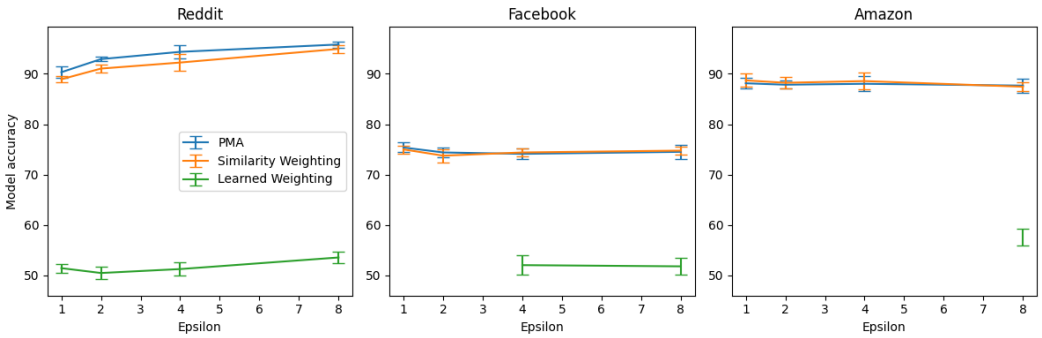

Figure 3: Test accuracies for each model run on the Reddit, FacebookPagePage, and Amazon datasets. Four epsilon values were tested ($\epsilon \in \{1, 2, 4, 8\}$) with delta fixed to $\delta = 10^{-5}$. For the Reddit dataset, all three models ran for all values of epsilon. For Facebook, the learned weighting mechanism was not able to train for 1000 iterations for $\epsilon \in \{1, 2\}$. In the Amazon dataset, the learned weighting mechanism could only train for 1000 iterations with $\epsilon = 8$.

models (the heuristic-weighted aggregation and PMA), we used all of the ϵ budget on aggregation perturbation. For all models, we used a classification module which would accept the zero-th and first hop node embeddings. These embeddings would pass to independent MLP respectively. The output of the MLP would be aggregated through concatenation and finally passed through a one layer "head" MLP. The head MLP predicts the node label. We used a cross-entropy loss function.

4.4 Results

Our results are summarized in Figure 3. The Reddit dataset was the largest but easiest. The similarity-weighted aggregation is competitive to GAP's PMA (unweighted) module but does not outperform it. The attention-weighted (Learned Weighting) aggregation performs significantly worse. It is clear that training a model by passing through the aggregation step for T iterations is not ideal. On the other hand, both GAP and our similarity weighting approach perform the aggregation step once to generate the cache.

On the FacebookPagePage dataset, the similarity-weighted aggregation is more competitive with the PMA mechanism. Surprisingly, the accuracy does not improve with an increased privacy budget. In fact, the average accuracy of both aggregation schemes reduces with an increase of ϵ from 1 to 2. This effect implies that the edge information is not important to predictions. On this dataset, we could not train the attention model for 1000 iterations for privacy budgets $\epsilon = 1, 2$. This is because we fixed the noise scale and number of iterations.

The Amazon dataset exemplifies a similar effect to the FacebookPagePage dataset. On this dataset, we could only train the attention model for 1000 iterations for $\epsilon = 8$. Unfortunately, due to time constraints, we only experimented with these datasets. Also, the chosen datasets are all transductive and social networks. Future investigations should experiment with different kinds of datasets and also with node classification in the inductive setting.

5 Conclusion

In this paper, we explored two different aggregation methods meant to improve upon GAP [13]. We started by identifying a potential improvement that could be made to GAP – that the aggregation of node features could be weighted. A learned, attention-based weighting scheme seemed plausible owing to the success of the attention mechanism in other, non-private graph learning settings. However, implementing and proving differential privacy bounds for this approach was difficult, and resulted in our method consuming a large portion of available privacy budget. To combat the downsides of our learned attention-based method, we hypothesized that a simpler, non-learned, similarity based weighting scheme would produce better results. Our experiments showed that this was true, but still could not surpass the unweighted mechanism employed by GAP. We believe that some promising future research directions are (i) modifying the training of the learned attention weighting mechanism

to reduce the spent privacy budget; (ii) relaxing the requirement that only one hop is performed in the aggregation phase; (iii) using other heuristic similarity-based weightings. The results presented in this work may be negative, but we showed that tweaking the weighting mechanism is a promising, though challenging, approach for improving DP-GNNs.

References

[1] O. Lange and L. Perez, "Traffic prediction with advanced graph neural networks," Sep 2020. [Online]. Available: https://www.deepmind.com/blog/traffic-prediction-with-advanced-graph-neural-networks

[2] F. Monti, F. Frasca, D. Eynard, D. Mannion, and M. M. Bronstein, "Fake news detection on social media using geometric deep learning," 2019. [Online]. Available: https://arxiv.org/abs/1902.06673

[3] C. Eksombatchai, P. Jindal, J. Z. Liu, Y. Liu, R. Sharma, C. Sugnet, M. Ulrich, and J. Leskovec, "Pixie: A system for recommending 3+ billion items to 200+ million users in real-time," 2017. [Online]. Available: https://arxiv.org/abs/1711.07601

[4] X. He, R. Wen, Y. Wu, M. Backes, Y. Shen, and Y. Zhang, "Node-level membership inference attacks against graph neural networks," 2021. [Online]. Available: https://arxiv.org/abs/2102.05429

[5] I. E. Olatunji, W. Nejdl, and M. Khosla, "Membership inference attack on graph neural networks," 2021. [Online]. Available: https://arxiv.org/abs/2101.06570

[6] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in Theory of Cryptography, S. Halevi and T. Rabin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 265–284.

[7] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in Advances in Cryptology - EUROCRYPT 2006, S. Vaudenay, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 486–503.

[8] C. Dwork, G. N. Rothblum, and S. Vadhan, "Boosting and differential privacy," in 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, 2010, pp. 51–60.

[9] P. Kairouz, S. Oh, and P. Viswanath, "The composition theorem for differential privacy," in Proceedings of the 32nd International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1376–1385. [Online]. Available: https://proceedings.mlr.press/v37/kairouz15.html

[10] I. Mironov, "Rényi differential privacy," in 2017 IEEE 30th Computer Security Foundations Symposium (CSF). IEEE, aug 2017. [Online]. Available: https://doi.org/10.1109%2Fcsf.2017.11

[11] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, oct 2016. [Online]. Available: https://doi.org/10.1145%2F2976749.2978318

[12] C. Waites, "Pyvacy: Towards practical differential privacy for deep learning," 2019.

[13] S. Sajadmanesh, A. S. Shamsabadi, A. Bellet, and D. Gatica-Perez, "Gap: Differentially private graph neural networks with aggregation perturbation," 2022. [Online]. Available: https://arxiv.org/abs/2203.00949

[14] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. B. Wiltschko, "A gentle introduction to graph neural networks," Distill, 2021, https://distill.pub/2021/gnn-intro. [Online]. Available: https://distill.pub/2021/gnn-intro/

[15] F. Wu, Y. Long, C. Zhang, and B. Li, "Linkteller: Recovering private edges from graph neural networks via influence analysis," 2021. [Online]. Available: https://arxiv.org/abs/2108.06504

[16] I. E. Olatunji, T. Funke, and M. Khosla, "Releasing graph neural networks with differential privacy guarantees," 2021. [Online]. Available: https://arxiv.org/abs/2109.08907

[17] A. Daigavane, G. Madan, A. Sinha, A. G. Thakurta, G. Aggarwal, and P. Jain, "Node-level differentially private graph neural networks," 2021. [Online]. Available: https://arxiv.org/abs/2111.15521

[18] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2017. [Online]. Available: https://arxiv.org/abs/1710.10903

[19] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016. [Online]. Available: https://arxiv.org/abs/1609.02907

[20] A. Kolluri, T. Baluta, B. Hooi, and P. Saxena, "Lpgnet: Link private graph networks for node classification," 2022. [Online]. Available: https://arxiv.org/abs/2205.03105

[21] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf

[22] S. Sajadmanesh and D. Gatica-Perez, "Locally private graph neural networks," 2020. [Online]. Available: https://arxiv.org/abs/2006.05535

[23] S. Hidano and T. Murakami, "Degree-preserving randomized response for graph neural networks under local differential privacy," 2022. [Online]. Available: https://arxiv.org/abs/2202.10209

[24] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2017. [Online]. Available: https://arxiv.org/abs/1706.02216

[25] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," 2018. [Online]. Available: https://arxiv.org/abs/1811.05868

[26] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," 2019. [Online]. Available: https://arxiv.org/abs/1909.13021

A Proofs

For the following proofs, let V be the set of all nodes, E be the set of all edges, and $\boldsymbol{X}$ be a node feature matrix where $\boldsymbol{X}_v$ corresponds to the features of node $v \in V$ as a row of matrix $\boldsymbol{X}$.

Lemma 1. *Let* $\texttt{agg}(\boldsymbol{X}, \boldsymbol{\alpha}) = \boldsymbol{\alpha}^T \boldsymbol{X}$ *be the weighted aggregation function. If the input feature matrix* $\boldsymbol{X}$ *is a row-normalized, such that* $\forall v \in V,\ \|\boldsymbol{X}_v\|_2 = 1$*, and the values of the weighted adjacency matrix* $\boldsymbol{\alpha}$ *are bounded between 0 and 1. Then, the edge-level sensitivity of the weighted aggregation function is* $\Delta\texttt{agg} = 1$.

Proof. The proof follows similarly to A.1 from [13].

Let $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha'}$ be attention-weighted adjacency matrices of two adjacent graphs differing by one edge (u, v). For these nodes u and v,

$$\begin{cases} \boldsymbol{\alpha}_{i,j} \neq \boldsymbol{\alpha'}_{i,j}, & i = u, j = v \\ \boldsymbol{\alpha}_{i,j} = \boldsymbol{\alpha'}_{i,j}, & \text{otherwise} \end{cases}$$

The sensitivity of $\texttt{agg}(\cdot)$ is given by,

$$\begin{aligned}\Delta\texttt{agg} &= \max\|\texttt{agg}(\boldsymbol{X}, \boldsymbol{\alpha}) - \texttt{agg}(\boldsymbol{X}, \boldsymbol{\alpha'})\|_F \\ &= \max\|\boldsymbol{\alpha}^T\boldsymbol{X} - \boldsymbol{\alpha'}^T\boldsymbol{X}\|_F \\ &= \max\left(\sum_{i\in V}\|\sum_{j\in V}(\boldsymbol{\alpha}_{j,i}\boldsymbol{X}_j - \boldsymbol{\alpha'}_{j,i}\boldsymbol{X}_j)\|_2^2\right)^{1/2}\end{aligned}$$

Since the adjacency matrices differ by one edge, we can simplify the sum to:

$$\max\|\boldsymbol{\alpha}_{u,v}\boldsymbol{X}_v - \boldsymbol{\alpha'}_{u,v}\boldsymbol{X}_v\|_2$$

We observe that this quantity is maximized when the differing edge (u, v) has a weight $\boldsymbol{\alpha}_{u,v} = 1$. Since this edge does not exist in the adjacent graph, its weight is $\boldsymbol{\alpha'}_{u,v} = 0$. This gives,

$$\|\boldsymbol{X}_v\|_2$$

Since $\boldsymbol{X}$ is row-normalized, $\|\boldsymbol{X}_v\|_2 = 1$. □

Theorem 1. *Assuming similarity metrics bounded between 0 and 1, one pass over the similarity-weighted aggregation has an edge-level privacy cost of* $(\alpha, K\alpha/2\sigma^2)$*-RDP.*

Proof. The heuristic-weighted aggregation uses a similarity function $\boldsymbol{\alpha}$ given by,

$$\boldsymbol{\alpha} = \begin{cases}\texttt{sigmoid}(\boldsymbol{X}_i\boldsymbol{X}_j^T), & (i,j) \in E \\ 0, & \text{otherwise}\end{cases}$$

Where $\texttt{sigmoid}(\cdot)$ is the sigmoid function. This similarity function is a weighted adjacency matrix that's values are bounded between 0 and 1. Gaussian noise is applied to the weighted aggregation function $\texttt{agg}(\boldsymbol{X}, \boldsymbol{\alpha}) = \boldsymbol{\alpha}^T\boldsymbol{X}$. According to Lemma 1, the edge-level sensitivity of this aggregation is 1. According to Corollary 3 of [10], each individual application of the Gaussian mechanism is $(\alpha, \alpha/2\sigma^2)$-RDP. Therefore, since the weighted aggregation uses K adaptive applications of the Gaussian mechanism, it must satisfy $(\alpha, K\alpha/2\sigma^2)$-RDP. □

Lemma 2. *Let* $a : (\boldsymbol{X}_{src}, \boldsymbol{X}_{dst}; \boldsymbol{\Theta}_{att}) \to \boldsymbol{e}$ *be the attention model with parameters* $\boldsymbol{\Theta}_{att}$ *that outputs attention coefficients* $\boldsymbol{e}$*. Assuming that for a batch we select a fixed set of nodes* $U \subseteq V$ *and all their incoming neighbours* N *satisfying* $\{n \in N \mid (n, u) \in E, n \in V, u \in U\}$*. If* $\boldsymbol{X}_{src}$ *corresponds to the features of the source node and* $\boldsymbol{X}_{dst}$ *corresponds to the features of destination node, for any edge* $(n, u) \in E \mid n \in N, u \in U$*. Then, model* a *satisfies the conditions of DP-SGD for edge-level DP.*

Proof. Let EL be $\{(n, u) \in E \mid n \in N, u \in U\}$. The attention mechanism is trained on the edges of EL. If a node u is selected to be in the batch, all of its incoming edges will be in EL. Therefore, the probability of a given edge to be in EL is equal to the probability of it's destination node being in U. This probability is fixed to $|U|/|V|$. This is enough to show that a satisfies the conditions of DP-SGD. □

Theorem 2. *The edge-level privacy cost of training the attention model is* $(\alpha, \epsilon_1(\alpha) + T\alpha/2\sigma^2)$*-RDP.*

Proof. The learned weighting aggregation uses a shared attention mechanism to learn attention coefficients that are bounded between 0 and 1 independently using the sigmoid non-linearity. If we use the batching method described in Lemma 2, then we can train the attention model using DP-SGD since the sensitivity of the gradients is bounded by clipping.

Suppose the attention model is trained for T iterations to satisfy $(\alpha, \epsilon_1(\alpha))$-RDP. It uses the attention coefficients to weight its aggregation function and applies the Gaussian mechanism to satisfy edge-level DP. According to Lemma 1, the edge-level sensitivity of this aggregation is 1. According to

Corollary 3 of [10], each individual application of the Gaussian mechanism is $(\alpha, \alpha/2\sigma^2)$-RDP. Since the Gaussian mechanism is applied adaptively every iteration for a total of T iterations, the aggregation must satisfy $(\alpha, T\alpha/2\sigma^2)$-RDP. As a result, the training is an adaptive composition of a $(\alpha, \epsilon_1(\alpha))$-RDP mechanism and a $(\alpha, T\alpha/2\sigma^2)$-RDP mechanism. Therefore, the total edge-level privacy cost for training is $(\alpha, \epsilon_1(\alpha) + T\alpha/2\sigma^2)$-RDP. □