Sparse but Strong: Crafting Adversarially Robust Graph Lottery Tickets

Subhajit Dutta Chowdhury[1*], **Zhiyu Ni**[1*], **Qingyuan Peng**[1*], **Souvik Kundu**[2], **Pierluigi Nuzzo**[1]
[1] University of Southern California, [2] Intel Labs

Abstract

Graph Lottery Tickets (GLTs), comprising a sparse adjacency matrix and a sparse graph neural network (GNN), can significantly reduce the inference latency and compute footprint compared to their dense counterparts. Despite these benefits, their performance against adversarial structure perturbations remains to be fully explored. In this work, we first investigate the resilience of GLTs against different structure perturbation attacks and observe that they are highly vulnerable and show a large drop in classification accuracy. Based on this observation, we then present an adversarially robust graph sparsification (ARGS) framework that prunes the adjacency matrix and the GNN weights by optimizing a novel loss function capturing the graph homophily property and information associated with both the true labels of the train nodes and the pseudo labels of the test nodes. By iteratively applying ARGS to prune both the perturbed graph adjacency matrix and the GNN model weights, we can find adversarially robust graph lottery tickets that are highly sparse yet achieve competitive performance under different untargeted training-time structure attacks. Evaluations conducted on various benchmarks, considering different poisoning structure attacks, namely, PGD, MetaAttack, Meta-PGD, and PR-BCD demonstrate that the GLTs generated by ARGS can significantly improve the robustness, even when subjected to high levels of sparsity.

1 Introduction

Graph neural networks (GNNs) [9, 14, 26, 32, 31] achieve state-of-the-art performance on various graph-based tasks like semi-supervised node classification [14, 9, 26], link prediction [29], and graph classification [28]. The success of GNNs is attributed to the neural message-passing scheme in which each node updates its feature by recursively aggregating and transforming the features of its neighbors. However, the effectiveness of GNNs, when scaled up to large and densely connected graphs, is adversely affected by the high training cost, high inference latency, and substantial memory consumption. Unified graph sparsification (UGS) [3] addresses this concern by simultaneously pruning the input graph adjacency matrix and the GNN to achieve a graph lottery ticket (GLT), a pair of sparse graph adjacency matrix and GNN model, which can potentially accelerate inference without compromising model performance.

Recent studies reveal that GNNs are vulnerable to adversarial attacks [4, 27, 37, 23, 12]. An adversarial attack introduces unnoticeable perturbations to the graph structure or node features. These perturbations increase the distribution shift between train nodes and test nodes, fooling the GNN to misclassify nodes in the graph [17]. When compared to node features, altering the graph structure has a more significant impact on the classification accuracy. To counter these attacks, many defense techniques have been developed that try to improve the classification accuracy of GNNs either by cleaning the perturbed graph structure [35, 30, 27, 13] or by introducing new learning approaches [17, 6]. On the other hand, while GLTs demonstrate strong performance on original

[*]Equal Contribution

NeurIPS 2023 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2023).

benign graph data, their performance in the presence of adversarial structure perturbations remains largely unexplored. *Finding adversarially robust GLTs is key to enabling efficient GNN inference under adversarial threats.*

To this end, we first empirically investigate the resilience of GLTs identified by UGS against different structure perturbation attacks [37, 20, 23]. Then, we present ARGS, adversarially robust graph sparsification, an end-to-end optimization framework that, given an adversarially perturbed graph, iteratively prunes the graph adjacency matrix and the GNN model weights to generate an adversarially robust graph lottery ticket (ARGLT) that achieves competitive classification accuracy while exhibiting high levels of sparsity. In this work, we consider a widely adopted setting for adversarial attacks, i.e., poisoning adversarial attacks on the graph structure. Attacks like the projected gradient descent (PGD) topology attack [27], the meta-learning-based graph attack (MetaAttack) [37], and Meta-PGD [23] often introduce many of the edge modifications around the training nodes [17] while the local structure of the test nodes is less affected. Moreover, the adversarial edges introduced are often between nodes with dissimilar features [27]. We leverage this information to formulate a new loss function that better guides the pruning of the adversarial edges in the graph and weights of the GNN. Additionally, we use self-learning to train pruned GNNs on sparse graph structures, which improves the classification accuracy of the GLTs.

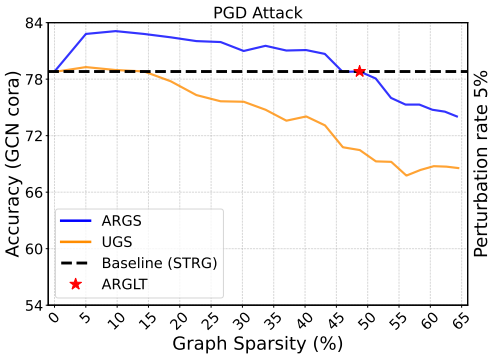

Figure 1: Comparison of different graph sparsification techniques in accuracy vs. graph sparsity trade-off plot. ARGS archives similar accuracy with 30% more sparsity compared to UGS for the Cora dataset under PGD attack.

Our proposal is experimentally verified across various GNN architectures on different graph datasets (Cora, Citeseer, PubMed, and OGBN-ArXiv) attacked by poisoning attacks (PGD, MetaAttack, Meta-PGD, PR-BCD [8]) for the node classification task. Results show that ARGS is widely applicable for sparsifying the GNN and the graph adjacency matrix when the graph structure has been adversarially attacked. By iteratively applying ARGS, ARGLTs can be broadly located across the 4 graph datasets with substantially reduced inference costs and unimpaired performance. The ARGLTs achieve $23\% - 61\%$ sparsity on graphs and $64\% - 98\%$ sparsity on GNN models, at little to no adversarial performance degradation. Figure 1 shows that, for node classification on Cora attacked by the PGD attack (5% perturbation), our ARGLT achieves similar accuracy to that of the full model and graph even with high graph and model sparsity of 49% and 95%, respectively. When compared to the GLTs identified by UGS, the ARGLTs on average achieve the same accuracy with $\sim 2.4\times$ more graph sparsity and $\sim 2.3\times$ more model sparsity.

2 Related Work

Graph Lottery Ticket Hypothesis. The lottery ticket hypothesis (LTH) [7] conjectures that there exist small sub-networks, dubbed as lottery tickets (LTs), within a dense randomly initialized network, that can be trained in isolation to achieve comparable accuracy to their dense counterparts. UGS made it possible to extend the LTH to GNNs [3], showing the existence of GLTs that can accelerate GNN inference. A GNN sub-network along with a sparse graph is defined as a GLT if the sub-network with the original initialization trained on the sparsified graph has a matching test accuracy to the original unpruned GNN trained on the full graph. Specifically, during end-to-end training, UGS applies two differentiable binary mask tensors to the graph adjacency matrix and the GNN model weights, respectively. After training, the lowest-magnitude elements are removed and the corresponding mask location is updated to 0, eliminating the low-scored edges and weights from the adjacency matrix and the GNN, respectively. The sparse GNN weight parameters are then rewound to their original initialization. To identify the GLTs, the UGS algorithm is applied in an iterative fashion until pre-defined graph and weight sparsity levels are reached. Experimental results show that UGS can significantly trim down the inference computational cost without compromising the predictive accuracy. In this work, we aim to find GLTs for datasets that have been adversarially perturbed. When

we apply the UGS algorithm directly to the perturbed graphs, the performance accuracy of the GLTs is substantially low compared to their clean counterparts, calling for new optimization method to find adversarially robust GLTs.

Adversarial Attacks on Graphs. Adversarial attacks on graphs can be classified as poisoning attacks, perturbing the graph at train time, and evasion attacks, perturbing the graph at test time. Both poisoning and evasion attacks can be targeted or global attacks [20]. A targeted attack deceives the model to misclassify a specific node [36, 1]. A global attack degrades the overall performance of the model [37, 27]. Depending on the amount of information available, the existing attacks can be categorized into white-box attacks, practical black-box attacks, and restricted black-box attacks [36, 2]. An attacker can modify the node features, the discrete graph structure, or both. Different attacks show that structure perturbation is more effective when compared to modifying the node features. Examples of global poisoning attacks include the MetaAttack [37], PGD attack [27], and PR-BCD attack [8]. Gradient-based attacks like PGD and MetaAttack treat the adjacency matrix as a parameter tensor and modify it via scaled gradient-based perturbations that aim to maximize the loss, thus resulting in degradation of the GNN prediction accuracy. PR-BCD [8] is a more scalable first-order optimization attack that can scale up to large datasets like OGBN-ArXiv [10]. Global poisoning attacks are highly effective in reducing the classification accuracy of different GNNs and are typically more challenging to counter since they modify the graph structure before training [34]. Therefore, we consider global graph structure poisoning attacks.

Defenses on Graphs. Several approaches for improving the robustness of GNNs have been developed to combat adversarial attacks on graphs [25, 5, 33, 13, 30, 27]. Many of these techniques try to improve the classification accuracy by preprocessing the graph structure, i.e., they detect the potential adversarial edges and assign these edges lower weights or even remove them. Jaccard-GCN [27] removes all edges between nodes whose features exhibit a Jaccard similarity below a certain threshold by leveraging the homophily property of graphs. SVD-GCN [5] replaces the adjacency matrix with a low-rank approximation since many real-world clean graphs are low-rank and attacks tend to disproportionately affect the high-frequency spectrum of the adjacency matrix. ProGNN [13] leverages low-rank, sparsity, and feature smoothness properties of graphs to clean the perturbed adjacency matrix and improve the trained GNN performance. GNNGuard [30] learns weights for the edges in each message passing aggregation via cosine-similarity and penalizes the adversarial edges by either filtering them or assigning less weight. Other techniques try to improve the GNN performance by enhancing model training through data augmentation [16, 6], adversarial training [27], self-learning [17], or by developing novel GNN layers, e.g., RGCN [33], which adopts Gaussian distributions as the hidden representations of the nodes in each convolutional layer to absorb the effect of an attack. Graph preprocessing tends to remove only a small fraction of edges from the adjacency matrix. Thus, the existing defense techniques often lead to high inference latency and are not scalable to real-world graphs. Differently from the existing defense methods that are built on dense GNNs with dense or nearly dense adjacency matrices, we aim to improve the robustness of low-latency sparse GNNs with highly sparse adjacency matrices. However, as robustness generally requires more non-zero parameters, yielding sufficiently sparse robust GLTs remains a challenge.

3 Methodology

Notations. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ represent an undirected graph with $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges. The topology of the graph can be represented with an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, where $\boldsymbol{A}_{ij} = 1$ if there is an edge $e_{i,j} \in \mathcal{E}$ between nodes v_i and v_j, while $\boldsymbol{A}_{ij} = 0$ otherwise. Each node $v_i \in \mathcal{V}$ has a feature vector $\mathbf{x}_i \in \mathbb{R}^F$, where F is the number of node features. Let $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}|\times F}$ and $\boldsymbol{Y} \in \mathbb{R}^{|\mathcal{V}|\times C}$ denote the feature matrix and the labels of all nodes in the graph, respectively, where C is the total number of classes. In this paper, we will also represent a graph as a pair $\{\boldsymbol{A}, \boldsymbol{X}\}$. In the case of message-passing GNN, the representation of a node v_i is iteratively updated by aggregating and transforming the representations of its neighbors. As an example, a two-layer[14] GNN can be specified as

$$\boldsymbol{Z} = f(\{\boldsymbol{A}, \boldsymbol{X}\}, \boldsymbol{\Theta}) = \mathcal{S}(\hat{\boldsymbol{A}}\sigma(\hat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}_{(0)})\boldsymbol{W}_{(1)}), \tag{1}$$

where $\boldsymbol{Z}$ is the prediction, $\boldsymbol{\Theta} = (\boldsymbol{W}_0, \boldsymbol{W}_1)$ are the weights, $\sigma(.)$ is the activation function, e.g., a rectified linear unit (ReLU), $\mathcal{S}(.)$ is the softmax function, $\hat{\boldsymbol{A}} = \tilde{\boldsymbol{D}}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\tilde{\boldsymbol{D}}^{-\frac{1}{2}}$ is the normalized adjacency matrix with self-loops, and $\tilde{\boldsymbol{D}}$ is the degree matrix of $\boldsymbol{A} + \boldsymbol{I}$. We consider the transductive

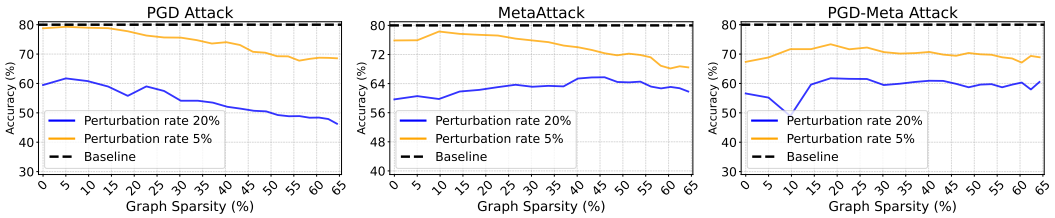

Figure 2: Classification accuracy of GLTs generated using UGS for the Cora dataset under PGD (left), MetaAttack (middle), and Meta-PGD attack (right) with 5% and 20% perturbation rates. The baseline refers to the accuracy on the clean graph.

semi-supervised node classification (SSNC) task for which the cross-entropy (CE) loss over labeled nodes is given by

$$\mathcal{L}_0(f(\{\boldsymbol{A}, \boldsymbol{X}\}, \boldsymbol{\Theta})) = -\sum_{l \in \mathcal{Y}_{TL}} \sum_{j=1}^{C} \boldsymbol{Y}_{l_j} \log(\boldsymbol{Z}_{l_j}), \tag{2}$$

where $\mathcal{Y}_{TL}$ is the set of train node indices, C is the total number of classes, and $\boldsymbol{Y}_l$ is the one-hot encoded label of node v_l.

Graph Lottery Tickets. A GLT consists of a sparsified graph, obtained by pruning some edges in $\mathcal{G}$, and a GNN sub-network, with the original initialization, that can be retrained to achieve comparable performance to the original GNN trained on the full graph, where performance is measured in terms of test accuracy. Given a GNN $f(\cdot, \boldsymbol{\Theta})$ and a graph $\mathcal{G} = \{\boldsymbol{A}, \boldsymbol{X}\}$, the associated GNN sub-network and the sparsified graph can be represented as $f(\cdot, \boldsymbol{m}_\theta \odot \boldsymbol{\Theta})$ and $\mathcal{G}_s = \{\boldsymbol{m}_g \odot \boldsymbol{A}, \boldsymbol{X}\}$, respectively, where $\boldsymbol{m}_g$ and $\boldsymbol{m}_\theta$ are differentiable masks applied to the adjacency matrix $\boldsymbol{A}$ and the model weights $\boldsymbol{\Theta}$, respectively, and $\odot$ is the element-wise product. UGS[3] finds the two masks $\boldsymbol{m}_g$ and $\boldsymbol{m}_\theta$ such that the sub-network $f(\cdot, \boldsymbol{m}_\theta \odot \boldsymbol{\Theta})$ along with the sparsified graph $\mathcal{G}_s$ can be trained to a similar accuracy as $f(., \boldsymbol{\Theta})$.

Poisoning Attack on Graphs. In this work, we investigate the robustness of GLTs under non-targeted poisoning attacks modifying the structure of the graph. In the case of a poisoning attack, GNNs are trained on a graph that attackers maliciously modify. The aim of the attacker is to find an optimal perturbed $\boldsymbol{A}^{'}$ that fools the GNN into making incorrect predictions. This can be formulated as a bi-level optimization problem [36, 37]:

$$\begin{aligned} &\arg\max_{\boldsymbol{A}^{'} \in \Phi(\boldsymbol{A})} \mathcal{L}_{atk}(f(\{\boldsymbol{A}^{'}, \boldsymbol{X}\}, \boldsymbol{\Theta}^*)) \\ \text{s.t.} \quad &\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} \mathcal{L}_0(f(\{\boldsymbol{A}^{'}, \boldsymbol{X}\}, \boldsymbol{\Theta})) \end{aligned} \tag{3}$$

where $\Phi(\boldsymbol{A})$ is the set of adjacency matrices that fit the constraint $\frac{||\boldsymbol{A}^{'} - \boldsymbol{A}||_0}{||\boldsymbol{A}||_0} \leq \Delta$, $\mathcal{L}_{atk}$ is the attack loss function, Δ is the perturbation rate, and $\boldsymbol{\Theta}^*$ is the optimal parameter for the GNN on the perturbed graph.

3.1 UGS Analysis Under Adversarial Attacks

We perform the MetaAttack [37], PGD [27], and Meta-PGD [23] attacks on the Cora dataset with different perturbation rates. Then, we apply UGS on these perturbed graphs to find the GLTs. As evident from Fig. 2, the classification accuracy of the GLTs identified by UGS is lower than the clean graph accuracy. The difference increases substantially when the perturbation rate increases. For example, in the PGD attack, when the graph sparsity is 30%, at 5% perturbation, the accuracy drop is 6%. This drop increases to 25% when the perturbation rate is 20%. Moreover, for 20% perturbation rate, even with 0% sparsity, the accuracy of the GNN is around 20% lower than that of the clean graph accuracy. While UGS removes edges from the perturbed adjacency matrix, it may not effectively remove the adversarially perturbed edges. A naïve application of UGS may not be sufficient to improve the adversarial robustness of the GLTs. Consequently, there is a need for an

adversarially robust UGS technique that can efficiently remove the edges affected by adversarial perturbations while pruning the adjacency matrix and the associated GNN, along with improved adversarial training, allowing the dual benefits of improved robustness and inference latency.

3.2 Analyzing the Impact of Adversarial Attacks on the Graph Properties

Adversarial attacks like MetaAttack, PGD, and PR-BCD poison the graph structure by either introducing new edges or deleting existing edges, resulting in changes in the original graph properties. We analyze the difference in the attribute features of the nodes connected by the clean and adversarial edges. Figure 3 depicts the density distribution of the attribute feature difference between connected nodes in the Citeseer graph dataset attacked by the PGD attack. We can observe from Figure 3 that the attack tends to connect nodes with large attribute feature differences. A defense technique can potentially leverage this information to differentiate between the benign and adversarial edges in the graph. ARGS uses this observation to iteratively prune the adversarial edges from homophilic graphs.

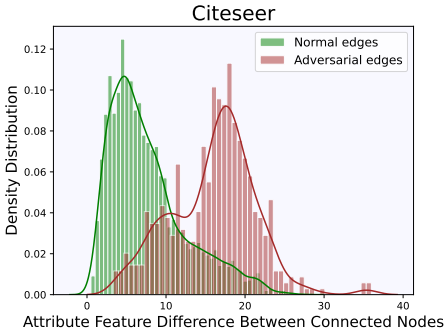

Figure 3: Density distribution of attribute feature differences between connected nodes in perturbed Citeseer graph dataset.

3.3 Adversarially Robust Graph Sparsification

We present ARGS, a sparsification technique that simultaneously reduces edges in $\mathcal{G}$ and GNN parameters in $\mathbf{\Theta}$ under adversarial attack conditions to effectively accelerate GNN inference yet maintain robust classification accuracy. ARGS reformulates the loss function to include (a) a CE loss term on the train nodes, (b) a CE loss term on a set of test nodes, and (c) a square loss term on all edges. Pruning the edges based on this combined loss function results in the removal of adversarial as well as less-important non-adversarial edges from the graph.

Removing Edges Around Train Nodes. Poisoning attacks like the MetaAttack and the PGD attack tend to modify more the local structure around the train nodes than that around the test nodes [17]. Specifically, a large portion of the modifications is introduced to the edges connecting a train node to a test node or a train node to another train node. We include a CE loss term associated with the train nodes, as defined in (2) in our objective function to account for the edges surrounding the train nodes. These edges include both adversarial and non-adversarial edges.

Removing Adversarial Edges. Adversarial attacks on graphs tend to connect nodes with distinct features [27]. However, it has been observed in numerous application domains involving social graphs, web page graphs, and citation graphs, that connected nodes within a graph tend to exhibit similar features [16, 22, 14]. To help remove the adversarial edges and encourage feature smoothness, we include the following loss to our objective function:

$$\mathcal{L}_{fs}(\boldsymbol{A}^{'}, \boldsymbol{X}) = \frac{1}{2} \sum_{i,j=1} \boldsymbol{A}^{'}_{ij} (\boldsymbol{x_i} - \boldsymbol{x_j})^2, \tag{4}$$

where $\boldsymbol{A}^{'}$ is the perturbed adjacency matrix and $(\boldsymbol{x}_i - \boldsymbol{x}_j)^2$ measures the feature difference between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Minimizing this loss function encourages removing edges that connect dissimilar nodes since they contribute a higher loss term.

Removing Edges Around Test Nodes. Removal of edges tends to be random in later iterations of UGS [11] since only a fraction of edges in $\mathcal{G}$ is related to the train nodes and directly impacts the corresponding CE loss. To better guide the edge removal around the test nodes, we also introduce a CE loss term for these nodes. However, the labels of the test nodes are unknown. We can then leverage the fact that structure poisoning attacks modify only the structure surrounding the train nodes, while their features and labels remain clean. Therefore, we first train a simple multi-layer perceptron (MLP) with 2 layers on the train nodes. MLPs only use the node features for training. We then use the trained MLP to predict the labels for the test nodes. We call these labels pseudo labels. Finally, we use the test nodes for which the MLP has high prediction confidence for computing the

Algorithm 1 Adversarially Robust Graph Sparsification

Input: Graph $\mathcal{G} = \{\boldsymbol{A}, \boldsymbol{X}\}$, GNN $f(\mathcal{G}, \boldsymbol{\Theta}_0)$ with initialization $\boldsymbol{\Theta}^0$, Sparsity levels s_g for graph and s_θ for GNN, Initial masks $\boldsymbol{m}_g = \boldsymbol{A}$, $\boldsymbol{m}_\theta = 1 \in \mathbb{R}^{||\boldsymbol{\Theta}_0||_0}$
Output: Final masks $\boldsymbol{m}_g$, $\boldsymbol{m}_\theta$

while $\left(1 - \frac{||\boldsymbol{m}_g||_0}{||\boldsymbol{A}||_0} < s_g\right)$ and $\left(1 - \frac{||\boldsymbol{m}_\theta||_0}{||\boldsymbol{\Theta}||_0} < s_\theta\right)$ **do**
  $\boldsymbol{m}_g^0 = \boldsymbol{m}_g$, $\boldsymbol{m}_\theta^0 = \boldsymbol{m}_\theta$, $\boldsymbol{\Theta}^0 = \{\boldsymbol{W}_0^0, \boldsymbol{W}_1^0\}$
  for $t = 0, 1, 2, \ldots, T-1$ **do**
    Forward $f(\cdot, \boldsymbol{m}_\theta^t \odot \boldsymbol{\Theta}^t)$ with $\mathcal{G} = \{\boldsymbol{m}_g^t \odot \boldsymbol{A}, \boldsymbol{X}\}$ to compute the loss $\mathcal{L}_{ARGS}$ in Equation 6
    $\boldsymbol{\Theta}^{t+1} \leftarrow \boldsymbol{\Theta}^t - \mu \nabla_{\boldsymbol{\Theta}^t} \mathcal{L}_{ARGS}$
    $\boldsymbol{m}_g^{t+1} \leftarrow \boldsymbol{m}_g^t - \omega_g \nabla_{\boldsymbol{m}_g^t} \mathcal{L}_{ARGS}$
    $\boldsymbol{m}_\theta^{t+1} \leftarrow \boldsymbol{m}_\theta^t - \omega_\theta \nabla_{\boldsymbol{m}_\theta^t} \mathcal{L}_{ARGS}$
  end for
  $\boldsymbol{m}_g = \boldsymbol{m}_g^{T-1}$, $\boldsymbol{m}_\theta = \boldsymbol{m}_\theta^{T-1}$
  Set percentage p_g of the lowest-scored values in $\boldsymbol{m}_g$ to 0 and set others to 1
  Set percentage p_θ of the lowest-scored values in $\boldsymbol{m}_\theta$ to 0 and set others to 1
end while

test node CE loss term. Let $\mathcal{Y}_{PL}$ be the set of test nodes for which the MLP prediction confidence is high and $\boldsymbol{Y}_{mlp}$ be the prediction by the MLP. The CE loss is given by

$$\mathcal{L}_1(f(\{\boldsymbol{A}^{'}, \boldsymbol{X}\}, \boldsymbol{\Theta})) = -\sum_{l \in \mathcal{Y}_{TL}} \sum_{j=1}^{C} \boldsymbol{Y}_{mlp_{l_j}} \log(\boldsymbol{Z}_{l_j}). \tag{5}$$

In summary, the complete loss function that ARGS optimizes is

$$\begin{aligned} \mathcal{L}_{ARGS} = \alpha \mathcal{L}_0(f(\{\boldsymbol{m}_g \odot \boldsymbol{A}^{'}, \boldsymbol{X}\}, \boldsymbol{m}_\theta \odot \boldsymbol{\Theta})) + \beta \mathcal{L}_{fs}(\boldsymbol{m}_g \odot \boldsymbol{A}^{'}, \boldsymbol{X}) \\ + \gamma \mathcal{L}_1(f(\{\boldsymbol{m}_g \odot \boldsymbol{A}^{'}, \boldsymbol{X}\}, \boldsymbol{m}_\theta \odot \boldsymbol{\Theta})) + \lambda_1 ||\boldsymbol{m}_g||_1 + \lambda_2 ||\boldsymbol{m}_\theta||_1, \end{aligned} \tag{6}$$

where β, γ, λ_1, and λ_2 are the hyperparameters. The value of α, γ is 1. λ_1 and λ_2 are the l_1 regularizers for $\boldsymbol{m}_g, \boldsymbol{m}_\theta$, respectively. After the training is complete, p_g, the lowest percentage of elements of $\boldsymbol{m}_g$, and p_θ, the lowest percentage of elements of $\boldsymbol{m}_\theta$, are set to 0. Then, the updated masks are applied to prune $\boldsymbol{A}$ and $\boldsymbol{\Theta}$, and the weights of the GNN are rewound to their original initialization value to generate the ARGLT.

We apply these steps iteratively until we reach the desired sparsity s_g and s_θ. Algorithm 1 illustrates our iterative pruning process yielding in ARGLTs. $||\cdot||_0$ is the L_0 norm counting the number of non-zero elements. As shown in Fig. 4 for the Cora dataset attacked by PGD attack with 20% perturbation, (a) most of the adversarial perturbation edges are between train and test nodes [17], and (b) our proposed sparsification technique successfully removes many of the adversarial edges. In particular, after applying our technique for 20 iterations, where each iteration removes 5% of the graph edges, the number of train-train, train-test, and test-test adversarial edges reduces by 68.13%, 47.3%, and 14.3%, respectively.

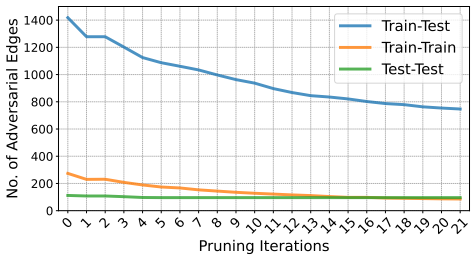

Figure 4: Evolution of adversarial edges in Cora dataset (attacked by PGD, 20% perturbation) as we apply ARGS to prune the graph. Train-Train edges connect two nodes from the train set. Train-Test edges connect two nodes from the train and test set, respectively. Test-Test edges connect two nodes from the test set.

Training Sparse ARGLTs. Structure poisoning attacks do not modify the labels of the nodes and the locality structure of the test nodes is less contaminated [17], implying that the train node labels and the local structure of the test nodes contain relatively "clean" information. We leverage this insight and train the GNN sub-network using both train nodes and test nodes. We use a CE loss term for both the train ($\mathcal{L}_0$) and test ($\mathcal{L}_1$) nodes. Since the true labels of the test nodes are not available, we train an MLP on the train nodes and then use it to predict the labels

for the test nodes [18, 17]. To compute the CE loss, we use only those test nodes for which the MLP has high prediction confidence. The loss function used for training the sparse GNN on the sparse adjacency matrix generated by ARGS is

$$\min_{\boldsymbol{\Theta}} \quad \eta \mathcal{L}_0(f(\{\boldsymbol{m}_g \odot \boldsymbol{A}^{'}, \boldsymbol{X}\}, \boldsymbol{m}_\theta \odot \boldsymbol{\Theta})) + \zeta \mathcal{L}_1(f(\{\boldsymbol{m}_g \odot \boldsymbol{A}^{'}, \boldsymbol{X}\}, \boldsymbol{m}_\theta \odot \boldsymbol{\Theta})) \tag{7}$$

where η, ζ are hyperparameters and $\boldsymbol{m}_\theta, \boldsymbol{m}_g$ are the masks evaluated by ARGS that are kept fixed throughout training. In the early pruning iterations, when graph sparsity is low, the test nodes are more useful in improving the model adversarial performance because the train nodes' localities are adversarially perturbed and there exist distribution shifts between the train and test nodes. However, as the graph sparsity increases, adversarial edges associated with the train nodes are gradually removed by ARGS, thus reducing the distribution shift and making the contribution of the remaining train nodes more important in the adversarial training.

4 Evaluation

Evaluation Setup. In this section, we validate the effectiveness of ARGS and the existence of adversarially robust GLTs across diverse graphs and GNN models under different adversarial attacks and perturbation rates. In particular, we evaluate our sparsification method on three widely used graph datasets, namely, Cora [21], Citeseer [24], and PubMed, which are attacked by three different structure poisoning attacks, namely, PGD [27], MetaAttack [37], and Meta-PGD [23] with different perturbation rates, i.e., $5\%, 10\%, 15\%, 20\%$. We instead attack the large-scale graph dataset OGBN-ArXiv [10] with a more scalable attack, called h projected randomized block coordinate descent (PR-BCD) attack [8]. We compare our method with UGS [3], and STRG [17]. For a fair comparison, we set $p_g = 5, p_\theta = 20$, similarly to the parameters used by UGS. More details on the dataset, model configurations, and hyperparameters in ARGS can be found in the appendix.

Analysis of ARGS. Figures 5 and 6 compare our method with UGS in terms of the test accuracy on the 2-layer GCN and GIN models, respectively. Here we consider the test classification accuracy of STRG as the baseline. We consider different perturbation rates for the different attacks. Figure 5 shows how the test accuracy of ARGS and UGS changes with the graph sparsity and the weight sparsity during the iterative pruning process on the three datasets Cora, Citeseer, and PubMed for GCN. In each pruning iteration, the graph is pruned by 5% while the GNN model weights are pruned by 20%. ARGLTs at a range of graph sparsity from 22.63% to 60.55% without performance deterioration can be identified across the GCN and GIN models. We can observe that the accuracy of our method is, on average, notably higher than the one of UGS. For example, in the case of the Cora dataset attacked by PGD with a perturbation budget of 15%, when the graph sparsity is 48.7% and the GNN weight sparsity is 94.61%, the accuracy of our technique is 77.92% whereas the accuracy of UGS is 55.84%. Therefore, for the same sparsity level, ARGS can achieve 22.08% better classification accuracy than UGS. Moreover, the higher accuracy of ARGS shows that the ARGLTs are more robust than those identified by UGS. In the case of the PubMed dataset, the accuracy of our method is stable even when the graph sparsity is high. This is due to the higher density of the PubMed graph allowing for more edges to be removed while maintaining accuracy. Similarly to GCN, ARGLTs can be identified by ARGS with high graph and model sparsity for GIN, as shown in Figure 6.

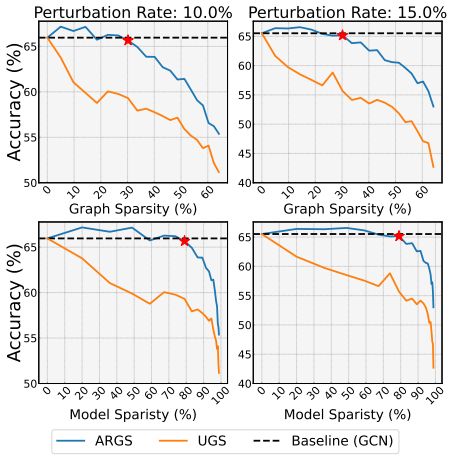

Figure 7: Node classification performance versus graph sparsity and model sparsity levels for GCN on OGBN-ArXiv dataset attacked by PR-BCD.

We also evaluate the robustness of ARGS on the large-scale dataset OGBN-ArXiv. We use the PR-BCD attack for perturbing the dataset and the reference GNN model is 28-layer ResGCN [15]. PGD or MetaAttack face timeout due to memory for these large graphs. Figure 7 shows that ARGS is able to identify ARGLTs that have high model and graph sparsity. In particular, the model sparsity

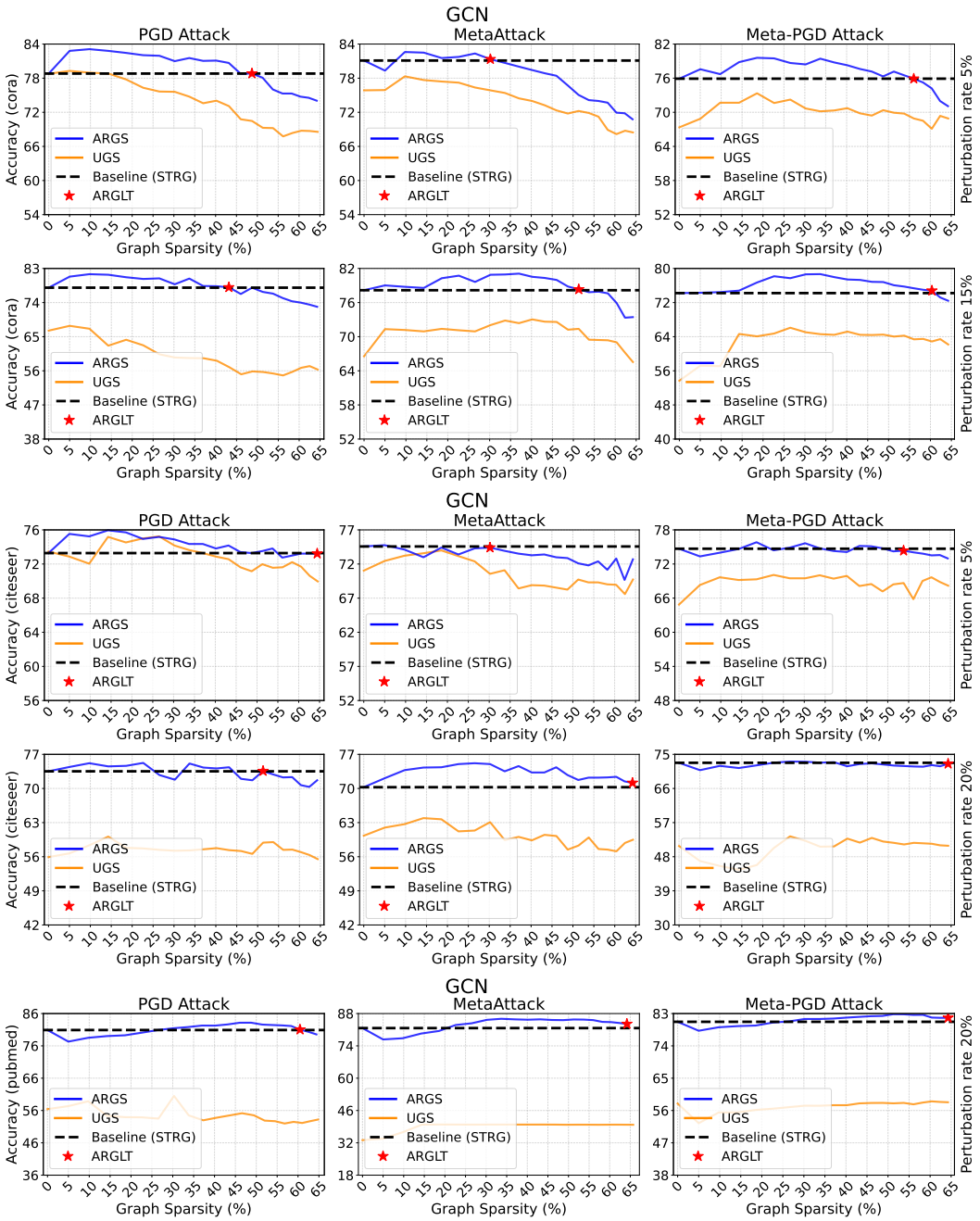

Figure 5: Node classification performance over achieved graph sparsity levels of GCNs on Cora, Citeseer, and PubMed datasets attacked by PGD, MetaAttack, and Meta-PGD with different perturbation rates. Red stars ⋆ indicate the ARGLTs which achieve similar performance with high sparsity. Dash black lines represent the classification accuracy of the baseline method STRG [17].

and graph sparsity are 79.03% and 30.17% for the 10% perturbed dataset, and 78.31% and 30.81%, respectively, for the 15% perturbed dataset. These results show that ARGS can find highly sparse GLTs also for large-scale graph datasets.

Ablation Study. To verify the effectiveness of each component of the proposed loss function used for the sparsification algorithm, we perform an ablation study as shown in Table 1. We consider the Cora dataset under PGD attack with 10% and 20% perturbation rates. Configuration 1 corresponds to

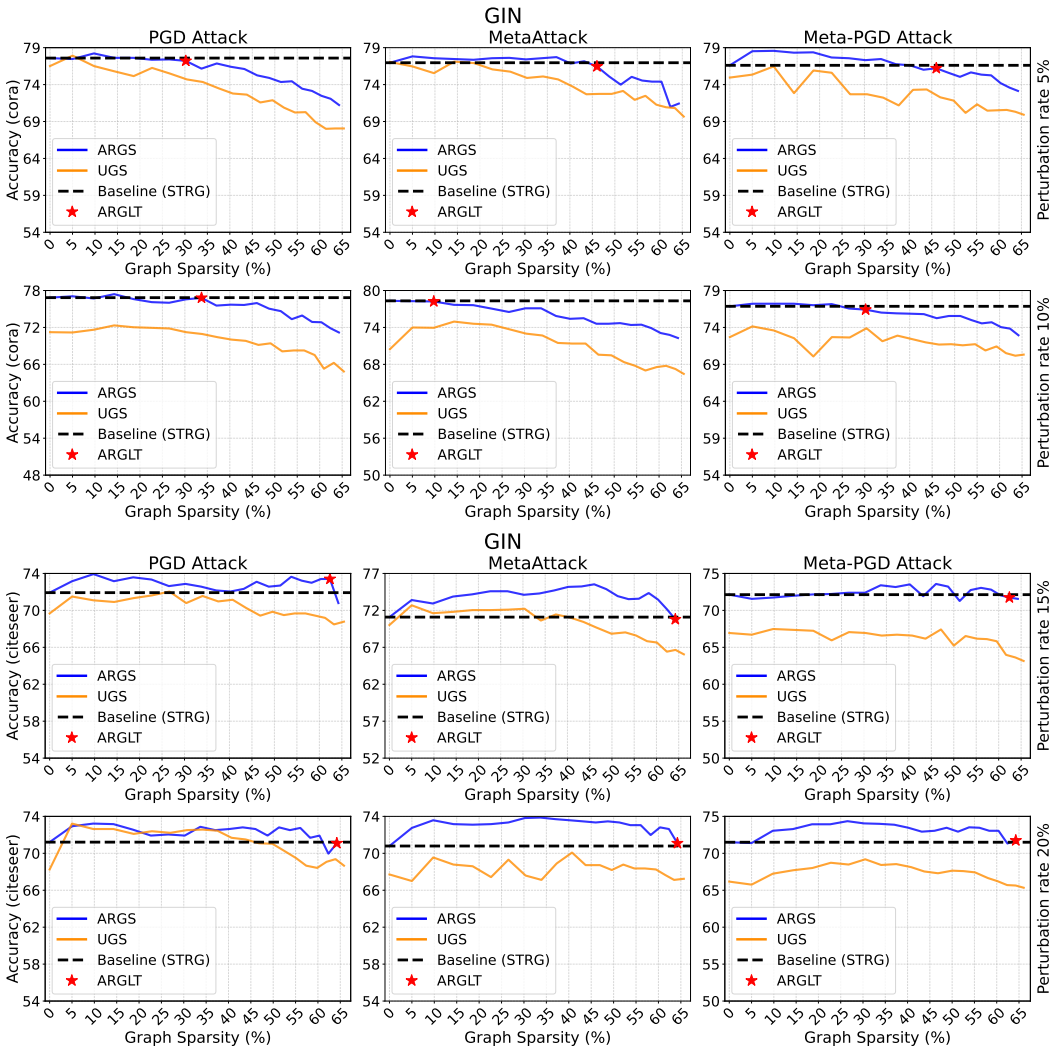

Figure 6: Node classification performance over achieved graph sparsity levels of GINs on Cora and Citeseer attacked by PGD, MetaAttack, and Meta-PGD with different perturbation rates.

ARGS with all the loss components in (6). In configuration 2, we do not use the feature smoothness component in (4) while performing the sparsification. In configuration 3, we skip the CE loss associated with the predicted test nodes in (5), and in configuration 4 we skip both the smoothness and CE loss on the predicted test nodes. As shown in Table 1, both configurations 2 and 3 improve the final performance compared to that of configuration 4, highlighting the importance of the losses introduced in (4) and (5). More importantly, at both high and low target sparsity, we yield the best classification performance with configuration 1, showcasing the importance of the unified loss function in (6). Further, ablation studies on different datasets and on the loss components associated with the loss function in (7) are provided in the appendix.

5 Conclusion

In this paper, we first empirically observed that the performance of GLTs collapses against structure perturbation poisoning attacks. To address this issue, we presented a new adversarially robust graph sparsification technique, ARGS, that prunes the perturbed adjacency matrix and the GNN weights by optimizing a novel loss function. By iteratively applying ARGS, we found ARGLTs that are highly sparse yet achieve competitive performance under different structure poisoning attacks. Our

Table 1: Ablation Study

GCN, Cora, PGD Attack						Classification Accuracy at Perturbation Rate 10%		Classification Accuracy at Perturbation Rate 20%	
Configuration	α	β	γ	η	ζ	Graph Sparsity 9.8% Model Sparsity 36.1%	Graph Sparsity 64.4% Model Sparsity 98.9%	Graph Sparsity 9.8% Model Sparsity 36.1%	Graph Sparsity 64.5% Model Sparsity 98.9%
1	✓	✓	✓	✓	✓	**83.25**	**75.10**	**80.63**	**75.60**
2	✓	✗	✓	✓	✓	82.04	70.57	78.92	64.84
3	✓	✓	✗	✓	✓	82.44	72.84	78.97	52.92
4	✓	✗	✗	✓	✓	80.58	62.42	75.7	54.18

evaluation showed the effectiveness of our method over UGS at both high and low-sparsity regimes. Future work involves the exploration of adversarially robust graph sparsification techniques in the case of graphs where the homophily property does not hold.

Acknowledgments

This research was supported in part by the National Science Foundation under Awards 1846524 and 2139982, the Office of Naval Research under Award N00014-20-1-2258, the Okawa Research Grant, and the USC Center for Autonomy and Artificial Intelligence.

References

[1] Aleksandar Bojchevski and Stephan Günnemann. Adversarial attacks on node embeddings via graph poisoning. In *International Conference on Machine Learning*, pages 695–704. PMLR, 2019.

[2] Heng Chang, Yu Rong, Tingyang Xu, Wenbing Huang, Honglei Zhang, Peng Cui, Wenwu Zhu, and Junzhou Huang. A restricted black-box adversarial framework towards attacking graph embedding models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3389–3396, 2020.

[3] Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A unified lottery ticket hypothesis for graph neural networks. In *International Conference on Machine Learning*, pages 1695–1706. PMLR, 2021.

[4] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International conference on machine learning*, pages 1115–1124. PMLR, 2018.

[5] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 169–177, 2020.

[6] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33:22092–22103, 2020.

[7] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[8] Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of graph neural networks at scale. *Advances in Neural Information Processing Systems*, 34:7637–7649, 2021.

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

[10] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[11] Bo Hui, Da Yan, Xiaolong Ma, and Wei-Shinn Ku. Rethinking graph lottery tickets: Graph sparsity matters. *arXiv preprint arXiv:2305.02190*, 2023.

[12] Wei Jin, Yaxin Li, Han Xu, Yiqi Wang, and Jiliang Tang. Adversarial attacks and defenses on graphs: A review and empirical study. *arXiv preprint arXiv:2003.00653*, 10(3447556.3447566), 2020.

[13] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 66–74, 2020.

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

[15] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020.

[16] Kuan Li, Yang Liu, Xiang Ao, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. Reliable representations make a stronger defender: Unsupervised structure refinement for robust gnn. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 925–935, 2022.

[17] Kuan Li, Yang Liu, Xiang Ao, and Qing He. Revisiting graph adversarial attack and defense from a data distribution perspective. In *The Eleventh International Conference on Learning Representations*, 2023.

[18] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[19] Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*, 2020.

[20] Xuanqing Liu, Si Si, Xiaojin Zhu, Yang Li, and Cho-Jui Hsieh. A unified framework for data poisoning attack to graph-based semi-supervised learning. *arXiv preprint arXiv:1910.14147*, 2019.

[21] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

[22] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

[23] Felix Mujkanovic, Simon Geisler, Stephan Günnemann, and Aleksandar Bojchevski. Are defenses for graph neural networks robust? *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 2022.

[24] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[25] Xianfeng Tang, Yandong Li, Yiwei Sun, Huaxiu Yao, Prasenjit Mitra, and Suhang Wang. Transferring robustness for graph neural network against poisoning attacks. In *Proceedings of the 13th international conference on web search and data mining*, pages 600–608, 2020.

[26] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[27] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples on graph data: Deep insights into attack and defense. *arXiv preprint arXiv:1903.01610*, 2019.

[28] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

[29] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[30] Xiang Zhang and Marinka Zitnik. Gnnguard: Defending graph neural networks against adversarial attacks. *Advances in neural information processing systems*, 33:9263–9275, 2020.

[31] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):249–270, 2020.

[32] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

[33] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1399–1407, 2019.

[34] Jiong Zhu, Junchen Jin, Michael T Schaub, and Danai Koutra. Improving robustness of graph neural networks with heterophily-inspired designs. *arXiv preprint arXiv:2106.07767*, 3, 2021.

[35] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. Deep graph structure learning for robust representations: A survey. *arXiv preprint arXiv:2103.03036*, 14, 2021.

[36] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2847–2856, 2018.

[37] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. 2019.

6 Appendix

In Section 6.1, we provide more details about the datasets used in the main sections of the paper. We describe the model and attack settings in Section 6.2. More experiments and ablation studies are presented in Section 6.4.

6.1 Dataset Details

We use three commonly used benchmark datasets, namely, Cora, Citeseer, PubMed, together with the OGBN-ArXiv to evaluate the efficacy of ARGS in finding ARGLTs. The details are summarized in Table 2. We use the data split $10\%/10\%/80\%$ (Train/Validation/Test) for these datasets and consider the largest connected component (LCC). For OGBN-ArXiv we follow the data split setting of Open Graph Benchmark (OGB) [10].

Table 2: Dataset details

Datasets	#Nodes	#Edges	Classes	Features
Cora	2485	5069	7	1433
Citeseer	2110	3668	6	3703
PubMed	19717	44338	3	500
OGBN-ArXiv	169343	1166243	40	128

6.2 Implementation Details

For a fair comparison, we follow the setup used by UGS as our default setting. For Cora, Citeseer, and PubMed we conduct all our experiments on two-layer GCN/GIN networks with 512 hidden units. The graph sparsity p_g and model sparsity p_θ are 5% and 20% unless otherwise stated. The value of β is chosen from $[0, 0.01, 0.1, 1]$ while the value of α, γ, η, and ζ is 1 by default. We use the Adam optimizer for training the GNNs. The value of λ_1 and λ_2 is 10^{-2}, 10^{-2} for Cora and Citeseer, while

for PubMed it is 1e-6, 1e-3, respectively. In each pruning round, the number of epochs to update the masks is by default 200 and we use early stopping. The 2-layer MLP used for predicting the pseudo labels of the test nodes has a hidden dimension of 1024. We use DeepRobust, an adversarial attack repository [19], to implement the PGD attack and MetaAttack on the Cora, Citeseer, and PubMed dataset for perturbation rates $5\%, 10\%, 15\%$ and 20%. In case of the PGD-Meta attack, we use the code provided by the authors to perform the attack [23]. We face OOM issues when applying these attacks on large-scale graph datasets. We use Pytorch-Geometric [10] for performing the PR-BCD attack on the OGBN-ArXiv dataset. An NVIDIA Tesla V100 32GB GPU is used to conduct all our experiments.

6.3 Ablation Study

We perform an ablation study to verify the effectiveness of each component of the proposed loss function used for the sparsification algorithm. A part of this ablation study is shown in Table 1. In this section, we present the evaluations performed on the Cora dataset for all the 3 different attacks with all the 4 different perturbation rates. Configuration 1 corresponds to ARGS executed with all the loss components. As shown in Table 3 and 4, at both high and low target sparsity, we yield the best classification performance with configuration 1, underscoring the importance of the selected loss function.

Table 3: Ablation Study

GCN, Cora, PGD Attack						Classification Accuracy at Perturbation Rate 5%		Classification Accuracy at Perturbation Rate 15%	
Configuration	α	β	γ	η	ζ	Graph Sparsity 22.7% Model Sparsity 67.7%	Graph Sparsity 60.4% Model Sparsity 98.2%	Graph Sparsity 22.7% Model Sparsity 67.7%	Graph Sparsity 60.4% Model Sparsity 98.2%
1	✓	✓	✓	✓	✓	**82.04**	**74.75**	**80.23**	**73.99**
2	✓	✗	✓	✓	✓	81.84	73.64	79.98	68.81
3	✓	✓	✗	✓	✓	81.69	74.45	76.86	72.89
4	✓	✗	✗	✓	✓	79.28	71.33	74.70	63.48
GCN, Cora, MetaAttack						Classification Accuracy at Perturbation Rate 5%		Classification Accuracy at Perturbation Rate 10%	
Configuration	α	β	γ	η	ζ	Graph Sparsity 22.7% Model Sparsity 67.6%	Graph Sparsity 62.3% Model Sparsity 98.6%	Graph Sparsity 22.6% Model Sparsity 67.5%	Graph Sparsity 64.2% Model Sparsity 98.9%
1	✓	✓	✓	✓	✓	**81.74**	**71.83**	**80.23**	**71.58**
2	✓	✗	✓	✓	✓	80.89	69.91	78.17	70.98
3	✓	✓	✗	✓	✓	79.88	71.33	75.40	66.81
4	✓	✗	✗	✓	✓	78.89	69.03	75.40	60.97

Table 4: Ablation Study

GCN, Cora, MetaAttack						Classification Accuracy at Perturbation Rate 15%		Classification Accuracy at Perturbation Rate 20%	
Configuration	α	β	γ	η	ζ	Graph Sparsity 22.7% Model Sparsity 67.6%	Graph Sparsity 60.4% Model Sparsity 98.2%	Graph Sparsity 22.6% Model Sparsity 67.5%	Graph Sparsity 60.3% Model Sparsity 98.2%
1	✓	✓	✓	✓	✓	**80.73**	**75.91**	**79.38**	**70.37**
2	✓	✗	✓	✓	✓	80.23	73.69	78.72	69.97
3	✓	✓	✗	✓	✓	77.97	72.74	75.50	69.16
4	✓	✗	✗	✓	✓	78.42	72.08	74.09	68.86
GCN, Cora, Meta-PGD Attack						Classification Accuracy at Perturbation Rate 5%		Classification Accuracy at Perturbation Rate 10%	
Configuration	α	β	γ	η	ζ	Graph Sparsity 22.7% Model Sparsity 67.7%	Graph Sparsity 60.4% Model Sparsity 98.2%	Graph Sparsity 22.7% Model Sparsity 67.6%	Graph Sparsity 60.3% Model Sparsity 98.2%
1	✓	✓	✓	✓	✓	**79.48**	**74.20**	**77.01**	**73.69**
2	✓	✗	✓	✓	✓	78.12	73.99	76.87	70.67
3	✓	✓	✗	✓	✓	76.86	69.67	76.01	69.97
4	✓	✗	✗	✓	✓	76.46	68.67	75.51	68.46
GCN, Cora, Meta-PGD Attack						Classification Accuracy at Perturbation Rate 15%		Classification Accuracy at Perturbation Rate 20%	
Configuration	α	β	γ	η	ζ	Graph Sparsity 22.7% Model Sparsity 67.4%	Graph Sparsity 60.4% Model Sparsity 98.2%	Graph Sparsity 22.7% Model Sparsity 67.4%	Graph Sparsity 60.4% Model Sparsity 98.2%
1	✓	✓	✓	✓	✓	**78.17**	**74.80**	**78.22**	**75.55**
2	✓	✗	✓	✓	✓	77.31	73.24	76.70	73.59
3	✓	✓	✗	✓	✓	77.26	70.88	77.46	67.25
4	✓	✗	✗	✓	✓	75.80	69.92	75.92	66.80

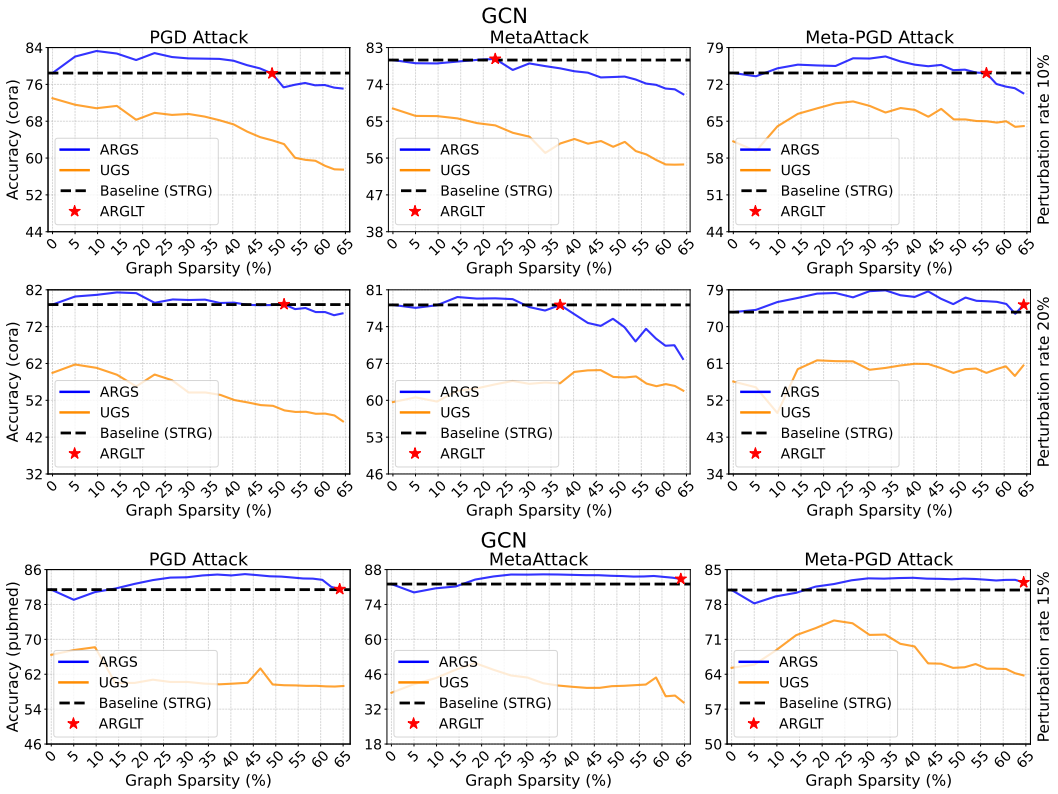

Figure 8: Node classification performance over achieved graph sparsity levels of GCNs on Cora and PubMed attacked by PGD, MetaAttack, and Meta-PGD with different perturbation rates.

6.4 Additional Results

Figure 8 shows the performance of ARGS in terms of the test accuracy on 2-layer GCN models. More specifically, in addition to the results with 5% and 15% perturbation rates with GCN for the 3 different attacks in Figure 5, we also include the results for 10% and 20% perturbation rates as well as the performance for the PubMed dataset.