Proceedings of Machine Learning Research 222, 2023 ACML 2023

Graph Structure Learning via Lottery Hypothesis at Scale

Yuxin Wang WANGYUXIN21@M.FUDAN.EDU.CN
School of Computer Science, Fudan University
Institute of Modern Languages and Linguistics, Fudan University

Xiannian Hu* 21210240194@M.FUDAN.EDU.CN
School of Computer Science, Fudan University

Jiaqing Xie* JIAXIE@ETHZ.CH
Department of Computer Science, ETH Zurich

Zhangyue Yin* YINZY21@M.FUDAN.EDU.CN
School of Computer Science, Fudan University

Yunhua Zhou 20110240059@FUDAN.EDU.CN
School of Computer Science, Fudan University

Xuanjing Huang XJHUANG@FUDAN.EDU.CN
School of Computer Science, Fudan University
Institute of Modern Languages and Linguistics, Fudan University

Xipeng Qiu† XPQIU@FUDAN.EDU.CN
School of Computer Science, Fudan University
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

Graph Neural Networks (GNNs) are commonly applied to analyze real-world graph-structured data. However, GNNs are sensitive to the given graph structure, which cast importance on graph structure learning to find optimal graph structures and representations. Previous methods have been restricted from large graphs due to high computational complexity. Lottery ticket hypothesis suggests that there exists a subnetwork that has comparable or better performance with proto-networks, which has been transferred to suit for pruning GNNs recently. There are few studies that address lottery ticket hypothesis's performance on defense in graphs. In this paper, we propose a scalable graph structure learning method leveraging lottery (ticket) hypothesis : GSL-LH. Our experiments show that GSL-LH can outperform its backbone model without attack and show better robustness against attack, achieving state-of-the-art performances in regular-size graphs compared to other graph structure learning methods without feature augmentation. In large graphs, GSL-LH can have comparable results with state-of-the-art defense methods other than graph structure learning, while bringing some insights into explanation of robustness.[1] [2] [3]

Keywords: Graph Neural Networks, Graph Structural Learning, Lottery Hypothesis, Large Scale Graph Learning

1. The code is available at: `https://github.com/jiaqingxie/GSL-LH`
2. ∗ Equal Contribution
3. † Corresponding author

1. Introduction

Graph Neural Networks (GNN) have demonstrated their efficiency and powerfulness in tasks including node classifications Hamilton et al. (2017); Kipf and Welling (2017), link predictions Schlichtkrull et al. (2018); Derr et al. (2018), graph classifications Xu et al. (2019b); Bianchi et al. (2021), and graph generations Kipf and Welling (2016); Jin et al. (2018). Effective message passing methods have been discovered by exploiting convolutional techniques Corso et al. (2020); Ma et al. (2020) or deep graph neural network architectures Li et al. (2020); Chiang et al. (2019). With the success of graph representation learning, a great deal of works explored into its related applications, including scenes such as knowledge graph completion Li et al. (2023), drug discovery Stärk et al. (2022), and point cloud generation Li et al. (2021a).

In the mentioned works above, graph structures are preserved throughout or before GNN training. However, graphs are easily attacked and GNN is vulnerable to attacks Wu et al. (2019); Dai et al. (2018); Zügner and Günnemann (2019). Removing edges from a fake news spreading network, for instance, might lead to worse misclassification results Karnyoto et al. (2022). The attack might in general convey less graph information than necessary, reducing GNN's capability of neighbouring node information aggregation which might be further aggravating node and graph level performance problems. It has been shown that graphs can be attacked directly by simply removing or injecting nosiy nodes during the training process Xu et al. (2019a); Wu et al. (2019), or in an indirect manner by training surrogate models and attacking those surrogate models Zügner et al. (2018); Zügner and Günnemann (2019), or in an RL approach by giving rewards when adding edges Ma et al. (2021b); Dai et al. (2018). We can verify whether the GNN structure is appropriate in order to design a better GNN to reduce the impact of structure changes via those attacking methods.

It is necessary to protect graphs from adversarial attacks, which could maintain the model's performance when perturbation rates are high. When graph or node features change significantly, *defense* methods aim to maintain representation and prediction levels. Adversarial based methods have been used as augmentations to defense attacks Goodfellow et al. (2014); Dai et al. (2018); Feng et al. (2019); Sun et al. (2019); Feng et al. (2019). A preprocessing method involved removing edges with low similarity scores from raw data Wu et al. (2019), namely graph purification. Previous methods of graph purification depended on graph regularization, which consumed a lot of computation when the graph was large. GNN cannot be trained on large graphs due to the scalability problem Chen et al. (2020a). Performing Singular Value Decomposition (SVD) on the adjacency matrices is an alternative method Entezari et al. (2020), but it does not ensure scalability since it cannot represent the graph densely despite selecting top-k features. Current graph structure learning (GSL) methods suffer from the same problem of non-scalability Jin et al. (2020). Since GSL is more intuitive and more explainable as shown in Figure 1, in this paper we focus on GSL as the backbone method to explore scalability.

When training deep neural networks, pruning methods are commonly used to deal with large amounts of parameters, for example masking nodes by setting their weights to zero. Lottery ticket hypothesis (**LTH**) Frankle and Carbin (2018), stated that training a subnetwork of the original dense network can achieve performance on par with that of training the original network with random initialization with fewer parameters. It is a

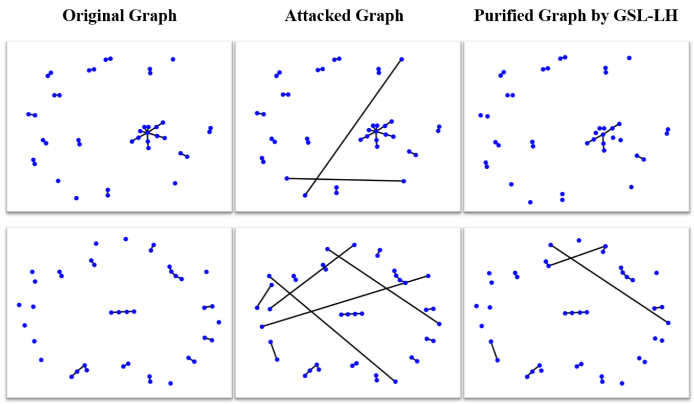

Figure 1: Visualization of two sub-graphs (original, attacked graph of perturbation rate 0.25, learned graph by GSL-LH) from Arxiv. The corresponding purified sub-graphs of GSL-LH in the third column are at 60% sparsity to make total edge numbers similar.

popular method for finding sparse and trainable neural networks in other fields. The graph adjacency matrix can also be thought of parameters from a view of lottery ticket hypothesis in Graph Neural Networks. Lottery ticket in Graph Neural Networks Chen et al. (2021b) was explored recently. Although Unified Graph Sparsification (UGS) Chen et al. (2021b) adopted the lottery ticket hypothesis, our work mainly focus on setting the purified graph as our destination instead of continuously pruning.

Our contributions could be summarized as follows:

- We propose graph structure learning (GSL) via lottery (ticket) hypothesis at scale for graph purifying. This method can be scaled up compared to previous GSL methods. This GSL method could be extended to large graphs.
- Our experiments show that GSL-LH can achieve state-of-the-art results of GSL methods without feature augmentation in regular-size graphs. In large graphs, GSL-LH can also improve backbone model's robustness and achieve comparable results with defense methods other than GSL.

2. Related Works

Lottery Ticket Hypothesis. Lottery Ticket Hypothesis (LTH) was first proposed in the computer vision field Frankle and Carbin (2018). Since then, finding sparse trainable subnetworks without performance degradation has been investigated in various machine learning fields such as computer vision Gan et al. (2021); Liu et al. (2019); Evci et al. (2019); You et al. (2020); Savarese et al. (2020); Gale et al. (2019); Chen et al. (2020b); Ma et al. (2021a); Renda et al. (2020), natural language processing Chen et al. (2020c); Yu et al. (2020), and continuous learning Chen et al. (2021c). Various architectures are available for backbone networks, including generative models Chen et al. (2021d,a); Kalibhat et al. (2020).

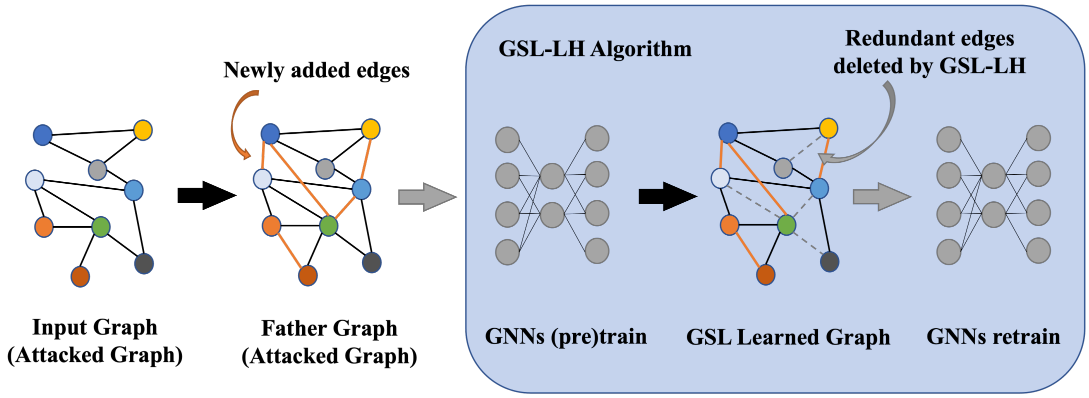

Figure 2: The training architecture of GSL-LH. Orange lines in father graph indicates edges newly added by sampling method. Dashed grey lines in GSL learned graph indicates redundant edges deleted by GSL-LH. GSL learned graph is got from lottery searching.

Iterative magnitude pruning (IMP) was proposed in the first work Frankle and Carbin (2018) to search for lotteries. Frankle et al. (2019b) modified IMP to achieve pruning in very early training, extreme sparsity and large-scale tasks. *LTH*'s scalability has also been investigated in other works Frankle et al. (2019a); Renda et al. (2020).

Graph Structure Learning for GNN. Graph Structure Learning (GSL) is one method for making Graph Neural Networks robust. At the same time as purifying the given graph structure, GSL learns how to represent the given data. Our work is in the field of **Direct Approaches** of GSL according to a recent survey Zhu et al. (2021). GCN-GT Yang et al. (2019) implements label smoothing in adjacency matrix learning, whereas feature smoothing is commonly used in feature learning. GLNN Gao et al. (2020) takes initial graph structures and feature smoothness into account, as well as adjacent sparsity, as part of a hybrid training objective. Pro-GNN Jin et al. (2020) assumes that a good adjacency matrix should be low-ranked so as to incorporate nuclear norm into training objective. Previously, GSL methods focused primarily on regular-size graphs. When applied to large graphs, they all require dense representations of adjacency matrices and are extremely computationally intensive. Geisler et al. (2021) proposed the first scalable attack method on large graphs, but few have developed defenses, let alone methods using GSL. As a result, our work is then focused on a scalable GSL method for training on large graphs.

3. Method

3.1. Graph Notations and Definitions

A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is undirected. $|\mathcal{V}|$ are the number of nodes. $|\mathcal{E}|$ are the number of edges. For each node v $\in \mathcal{V}$, it owns a corresponding feature vector $\mathbf{x} \in \mathbb{R}^F$, where the whole feature matrix of the graph can be expressed as $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}| \times \mathrm{F}}$. $\mathcal{E} = \{e_1, ..., e_{|\mathcal{E}|}\}$ is the edge set, where $e_n = (v_i, v_j) \in \mathcal{E}$ means that an edge is created by nodes v_i and v_j. The graph also has a topological property of an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$. If the edge $(v_i, v_j) \in \mathcal{E}$, then we can indicate from the matrix that the entry $\boldsymbol{A}_{i,j}$ is equal to 1, else the entry $\boldsymbol{A}_{i,j}$ is equal to 0.

3.2. Graph Convolution Networks

Based on Graph Convolution Networks (GCN), we revisit concepts related to GCN in the message passing model. Based on ChebNet, GCN Kipf and Welling (2017) simplified the Chebyshev Polynomials by taking the order as 1 and largest eigenvalue as 2.:

$$f_\theta * x = (\mathbf{I}_n + \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-\frac{1}{2}})\mathbf{X}\boldsymbol{\Theta} \tag{1}$$

In our baseline, we include the GCN model as the most basic model. A two-layer GCN block was included in the actual neural network training architecture. The prediction is as follows:

$$\mathbf{Z} = \mathcal{S}\left(\hat{\mathbf{A}}\sigma\left(\hat{\mathbf{A}}\mathbf{X}\boldsymbol{\Theta}^{\mathbf{1}}\right)\boldsymbol{\Theta}^{\mathbf{2}}\right) \tag{2}$$

where $X \in R^{N\times E}$ is the feature matrix of input nodes. $\boldsymbol{\Theta}^{\mathbf{1}}$ and $\boldsymbol{\Theta}^{\mathbf{2}}$ are weights assigned to each layer of a GNN. $\hat{\mathbf{A}}$ refers to the normalized adjacency matrix that has been discussed previously. Assume that only one block is used for training, then $\boldsymbol{Z}$ is the output of the graph neural network model $f(\mathcal{G}, \boldsymbol{\Theta})$. Otherwise, when the number of layers is larger than two, $\boldsymbol{Z}$ is regarded as the latent graph embeddings. $\mathcal{S}(\cdot)$ can be a softmax function (if multi-class classification) or a sigmoid function (if binary classification) , or simply a linear layer (non-linear regression). $\sigma(\cdot)$ denotes the activation function such as ReLU and LeakyReLU (GAT specified).

GSL Problem Restatement & Node Classification Problem. For node classification problem, $\mathcal{L}_{\text{CE}}(\boldsymbol{\Theta}, \mathbf{A}_F) := \sum_{i,c} y_{i,c} \log f(\boldsymbol{\Theta}, \mathbf{A}_F; x_i)_c$, where i represents an index of a given training sample and c represents the class of the corresponding training sample. Given an undirected graph $\mathcal{G}_0 = \{\mathcal{V}_0, \mathcal{E}_0\}$ as the input graph (or the attacked graph), graph purification is to find another graph $\mathcal{G}_* = \{\mathcal{V}_*, \mathcal{E}_*\}$ where $\mathcal{G}_*$ can better express the data structure of input data (or purify the perturbations).

3.3. Main algorithm: GSL-LH

Applying Lottery Ticket Hypothesis We apply Lottery Ticket Hypothesis (*LTH*) to search for graph $\mathcal{G}_*$. First we need to find the father graph of $\mathcal{G}_*$ to start lottery ticket searching. Only $\mathcal{G}_0$ is available, so we search from the father graph of $\mathcal{G}_0$, named as $\mathcal{G}_F$. To avoid making the searching space too large, we should carefully choose $\mathcal{G}_F$. A K-order neighbor of node i refers to the node j that the distance between i, j equals to K. For a graph adding all i-order neighbors ($i \in [1, K]$) to the adjacency matrix, we call it the K-order father graph. In regular-size graphs, we use K-order father graph of $\mathcal{G}_0$ to be our $\mathcal{G}_F$ to start searching lottery tickets. The calculation of K-order graphs in large graphs can be time-consuming, and even 2-order graphs can have tremendous edges, making GNNs inapplicable. Therefore we use lottery sampling to tackle this problem.

Lottery Sampling In large graphs, to further shrink the searching space, we apply sampling methods to filter new-added neighbors. We examine three easy sampling methods in this

Algorithm 1 GSL-LH algorithms

Input: Feature $\boldsymbol{X}$, label $\boldsymbol{y}$, input father graph $\mathcal{G}_F$, graph neural network $f(\boldsymbol{\Theta}, \mathbf{A}; \cdot)$, adjacent mask logits $\boldsymbol{\pi_{adj}}$, weight mask logits $\boldsymbol{\pi_{\Theta}}$, adjacent sparsity s_{adj}, weight sparsity s_{Θ}, number of steps for model and lottery search training N_1, N_2.

Stage 1: full model (pre-) train
Get model initialization $\boldsymbol{\Theta}_0$, $\mathbf{A}_F$.
for n=1 **to** N_1 **do**
  Update f with $\mathcal{L}_{\text{CE}}(\boldsymbol{\Theta}, \mathbf{A}_F) := \sum_{i,c} y_{i,c} \log f(\boldsymbol{\Theta}, \mathbf{A}_F; x_i)_c$.
end for
Stage 2: subgraph and subnetwork lottery searching
Initialize weight mask logits $\boldsymbol{\pi_{\Theta}}$ and adjacent mask logits $\boldsymbol{\pi_{adj}}$ to $\mathbf{1}$.
for n=1 **to** N_2 **do**
  Update module $\boldsymbol{\pi_{\Theta}}$ and $\boldsymbol{\pi_{adj}}$ with
    $\mathcal{L}_{GSL} = \mathcal{L}_{\text{CE}}(\boldsymbol{\pi_{adj}} \odot \boldsymbol{A_F}, \boldsymbol{\pi_{\Theta}} \odot \boldsymbol{\Theta_*})$.
end for
Stage 3: subgraph and subnetwork retrain
Obtain the module by pruning with sparsity s_{adj} and s_{Θ}.
    $\boldsymbol{m}_{adj} = \text{Prune}\left(\boldsymbol{\pi_{adj}}, s_{adj}\right)$
    $\boldsymbol{m}_{\Theta} = \text{Prune}\left(\boldsymbol{\pi_{\Theta}}, s_{\Theta}\right)$.
Set model parameters back to $\boldsymbol{\Theta}_0$.
for n=1 **to** N_1 **do**
  Update f with $\mathcal{L}_{\text{CE}}(\boldsymbol{m}_{adj} \odot \boldsymbol{A_F}, \boldsymbol{m}_{\Theta} \odot \boldsymbol{\Theta}_0)$.
end for

paper: random sampling, feature sampling, and lottery sampling. All of them need a number of sampling steps first, indicating the number of new-added neighbors for each node. Random sampling is to sample new neighbors randomly in all nodes. Feature sampling samples new neighbors by setting a threshold of similarities between node pairs. Lottery sampling is motivated by lottery distribution in operating systems. Since node features are available, we utilize it to help with sampling. First, we calculate the feature products between all pairs in the graph. For nodes i, j, $\boldsymbol{v_i}$ and $\boldsymbol{v_j}$ are their feature vectors, the attention matrix $\boldsymbol{P}$ containing their feature product scores is given by: $\boldsymbol{P}(i,j) = \boldsymbol{v_i}^T \boldsymbol{v_j}$, For one given node i, we normalize all nodes' scores by Softmax :

$$\boldsymbol{P}(i,j) = \frac{\exp(\boldsymbol{P}(i,j))}{\sum_{k=1}^{N} \exp(\boldsymbol{P}(i,k))} \tag{3}$$

where N is the node number of the graph. This operation seems to be highly computationally complex. However, for graphs like ogbn-Arxiv, this operation is fast enough on cpu and requires acceptable memories. Also, we only do this operation once for every dataset, and the answer matrix is portable, which makes the computational cost less important.

Next, we sample new neighbors of node i using $\boldsymbol{P}(i)$. For any node j, the lottery or the interval L_i it maintained is $\left(\sum_{k=1}^{j-1} \boldsymbol{P}(i,k), \sum_{k=1}^{j} \boldsymbol{P}(i,k)\right]$. For $j = 1$, it's $\left(0, \sum_{k=1}^{j} \boldsymbol{P}(i,k)\right]$. The sampling method starts from setting the sampling steps Ln for every node. For every sample step, we generate a random float number r in the uniform distribution U(0,1) and see in which interval it drops. If $r \in \left(\sum_{k=1}^{j-1} \boldsymbol{P}(i,k), \sum_{k=1}^{j} \boldsymbol{P}(i,k)\right]$, we add node j as a new neighbor to node i. We do this for Ln times and get father graph $\mathcal{G}_F$ of large graphs.

Graph Lottery Searching Given the father graph $\mathcal{G}_F$, we can start searching graph lottery via a subnetwork probing method Zhang et al. (2021). We propose two masks $\boldsymbol{m}_g$ and $\boldsymbol{m}_\Theta$ to respectively mask adjacency matrix and model weights. Specifically, the sparsity of $\boldsymbol{m}_g$ is equivalent to the sparsity of the original non-zero logits instead of the sparsity of the whole adjacency matrix. In large graphs where $\boldsymbol{A}$ is represented by edge index, $\boldsymbol{m}_g$ is also the size of original edge number which represents the edge value in practice.

First we train GNNs as normal to find a good weight $\boldsymbol{\Theta}_*$ using father graph $\mathcal{G}_F$. In the searching progress, the training objective is :

$$\mathcal{L}_{GSL} = \mathcal{L}_{\mathrm{CE}}(\boldsymbol{\pi}_g \odot \boldsymbol{A}_{\boldsymbol{F}}, \boldsymbol{\pi}_\Theta \odot \boldsymbol{\Theta}_*), \tag{4}$$

where $\boldsymbol{A}_{\boldsymbol{F}}$ is the adjacency matrix of $\mathcal{G}_F$ and $\boldsymbol{\Theta}$ is the weight of backbone model. $\odot$ means element-wise product. After lottery searching, we use the preset sparsity s_g and s_Θ to prune the mask $\boldsymbol{\pi}_g$ and $\boldsymbol{\pi}_\Theta$. Finally we bring our model back to its initialization $\boldsymbol{\Theta}_{\boldsymbol{0}}$ and retrain the model using pruned mask $\boldsymbol{m}_g$ and $\boldsymbol{m}_\Theta$. Algorithm 1 outlines our graph lottery searching framework.

	Nodes	**Edges**	**Classes**	**Features**
Cora	2,485	5,069	7	1,433
Cora ML	2,810	7,981	7	2,879
Citeseer	2,110	3,668	6	3,703
Pubmed	19,717	44,338	3	500
Arxiv	169,343	1,166,243	40	128

Table 1: Dataset Statistics.

Time Complexity Overall time spent on pretraining the full model is equivalent to training a backbone GNN model, as well as the time during subgraph lottery searching and subgraph network retraining. It shows a linear time complexity with regard to number of edges according to Wu et al. (2020), which is O(E) to GCN backbone for example. The time for pruning and masking is in linear time O(E). Therefore the overall time complexity of the purposed algorithm is subject to the triple training time of our graph neural networks plus the pruning time, which leads to O(4E) = O(E). The algorithm did improve the time complexity a lot when it compares to the complexity of SVD decomposition of previous methods mentioned, computationally large for large graph datasets. It generally explains why matrix decomposition oriented methods could not be trained efficiently on large graphs which is proved in our experiments.

4. Experiments

4.1. Datasets

We have evaluated GSL-LH on four regular-size graph datasets: Cora, Citeseer, Pubmed Kipf and Welling (2017), Cora ML Bojchevski and Günnemann (2018) and a large graph dataset:

ogbn-Arxiv Hu et al. (2020). The statistics of all datasets are listed in Table 1. For regular-size graph datasets, we adopt the experiment setting of Pro-GNN Jin et al. (2020). We split nodes into three parts, 10% for training, 10% for validation and 80% for test. For ogbn-Arxiv, we use the published open setting of OGB Hu et al. (2020). The paper nodes are splitted into three parts, training on papers published until 2017, validating on those published in 2018, and testing on those published since 2019. The attacked graphs of regular-size graphs are from DeepRobust Li et al. (2021b) using metattack Zügner and Günnemann (2019). The attacked graphs of ogbn-Arxiv are created by PR-BCD Geisler et al. (2021). We vary the perturbation rate (or the ratio of changed edges) from 5% to 25% at a step of 5% in attack.

4.2. Implementations and Baselines

We use a common two-layer GCN as the backbone encoder of GSL-LH. To make fair comparisons, for regular-size graphs, we set the hidden size to 16 to be the same as Pro-GNN settings. For ogbn-Arxiv, we set the hidden size to 128 following Geisler et al. (2021). We run all experiments on one Nvidia GeForce RTX 3090 GPU. As part of the training and retraining process, hyperparameters are fixed. During the lottery searching step, we only search for different learning rates for adjacency matrix masks and weight masks. Adjacent sparsity and weight sparsity are also hyperparameters. For more details, please refer to our supplemented materials. We conduct all the experiments 10 times and report the average accuracy with standard deviation.

We choose six different baselines including GCN Kipf and Welling (2017), GAT Veličković et al. (2018), RGCN Schlichtkrull et al. (2018), GCN-Jaccard Wu et al. (2019), GCN-SVD Entezari et al. (2020), Pro-GNN-fs Jin et al. (2020), PPRGO-based Bojchevski et al. (2020) methods. RGCN, GCN-Jaccard, and GCN-SVD focus on graph purifying while PPRGO Bojchevski et al. (2020) is based on pagerank.

Here we view GAT as a defense type of adaptive aggregation instead of graph purifying according to Deeprobust Li et al. (2021b). Since Pro-GNN-fs uses node features to intensify structure learning, we choose it over Pro-GNN.

4.3. Performances and Discussions

In Table 2, we observe that GSL-LH can outperform its backbone encoder GCN on all datasets without attack. Under attack, GSL-LH can improve the robustness of its backbone GCN on a large extent. Compared to other graph structure learning methods, GSL-LH can outperform most of them including GAT, GCN-SVD, GCN-Jaccard, RGCN. Moreover, GSL-LH achieves better performances than previous state-of-the-art graph structure learning method Pro-GNN-fs in Cora, Cora ML and Citeseer, but slightly lower performances in Pubmed, demonstrating the state-of-art performances of GSL-LH without feature augmentation. All mentioned graph structure learning methods fail to be applicable on Arxiv because of the usage of dense representation of adjacency matrix.

Since there are no applicable graph structure learning baselines on Arxiv, we compare our method with defense methods of other types on Arxiv. Geisler et al. (2021) proposed the first attack method on large graphs with defense methods Softmedian based on PPRGO. GSL-LH can also be applied to PPRGO by easily change the input adjacency matrix with our learned adjacency matrix. Results are listed in Table 3. Experiments have shown that

Dataset	Ptb Rate (%)	GCN	GAT	RGCN	GCN-Jaccard	GCN-SVD	Pro-GNN-fs	GSL-LH
Cora	0	83.50±0.44	83.97±0.65	83.09±0.44	82.05±0.51	80.63±0.45	83.42±0.52	**84.33±0.27**
	5	76.55±0.79	80.44±0.74	77.42±0.39	79.13±0.59	78.39±0.54	**82.78±0.39**	82.00±0.27
	10	70.39±1.28	75.61±0.59	72.22±0.38	75.16±0.76	71.47±0.83	77.91±0.86	**79.31±0.49**
	15	65.10±0.71	69.78±1.28	66.82±0.39	71.03±0.64	66.69±1.18	76.01±1.12	**77.95±0.40**
	20	59.56±2.72	59.94±0.92	59.27±0.37	65.71±0.89	58.94±1.13	68.78±5.84	**74.97±0.31**
	25	47.53±1.96	54.78±0.74	50.51±0.78	60.82±1.08	52.06±1.19	56.54±2.58	**68.92±0.60**
Cora ML	0	85.25±0.33	85.49±0.24	86.48±0.16	84.82±0.27	80.97±0.31	85.06±0.33	**86.50±0.17**
	5	79.19±0.36	81.06±0.59	81.62±0.14	80.23±0.40	80.23±0.34	83.25±0.71	**85.95±0.15**
	10	73.83±0.45	76.26±0.99	74.46±0.18	75.21±0.23	80.61±0.37	81.52±0.93	**84.78±0.10**
	15	54.35±0.66	57.96±1.34	54.87±0.33	57.45±0.65	**73.54±0.43**	53.81±0.27	71.66±0.30
	20	43.11±3.30	42.69±1.21	46.62±0.66	45.77±1.30	46.94±1.69	47.54±0.37	**70.37±1.70**
	25	48.45±0.48	43.74±4.44	50.15±0.36	49.05±0.41	56.28±0.86	50.99±0.27	**71.37±0.74**
Citeseer	0	71.96±0.55	73.26±0.83	71.20±0.83	72.10±0.63	70.65±0.32	73.26±0.38	**76.78±0.29**
	5	70.88±0.62	72.89±0.83	70.50±0.43	70.51±0.97	68.84±0.72	73.09±0.34	**74.96±0.25**
	10	67.55±0.89	70.63±0.48	67.71±0.30	69.54±0.56	68.87±0.62	72.43±0.52	**74.37±0.31**
	15	64.52±1.11	69.02±1.09	65.69±0.37	65.95±0.94	63.26±0.96	70.82±0.87	**71.73±0.56**
	20	62.03±3.49	61.04±1.52	62.49±1.22	59.30±1.40	58.55±1.09	66.19±2.38	**66.26±0.60**
	25	56.94±2.09	61.85±1.12	55.35±0.66	59.89±1.47	57.18±1.87	66.40±2.57	**66.77±0.37**
Pubmed	0	87.19±0.09	83.73±0.40	86.16±0.18	87.06±0.06	83.44±0.21	87.33±0.18	**87.46±0.05**
	5	83.09±0.13	78.00±0.44	81.08±0.20	86.39±0.06	83.41±0.15	87.25±0.09	**87.27±0.05**
	10	81.21±0.09	74.93±0.38	77.51±0.27	85.70±0.07	83.27±0.21	**87.25±0.09**	87.18±0.06
	15	78.66±0.12	71.13±0.51	73.91±0.25	84.76±0.08	83.10±0.18	**87.20±0.09**	86.90±0.03
	20	77.35±0.19	68.21±0.96	71.18±0.31	83.88±0.05	83.01±0.22	**87.09±0.10**	86.61±0.05
	25	75.50±0.17	65.41±0.77	67.95±0.15	83.66±0.06	82.72±0.18	**86.71±0.09**	86.37±0.07
Arxiv	0	71.03±0.27	**71.18±0.11**	-	-	-	-	70.90±0.20
	5	53.78±0.23	**55.74±0.27**	-	-	-	-	54.56±0.49
	10	46.83±0.15	48.68±0.33	-	-	-	-	**50.12±0.67**
	15	42.68±0.23	44.34±0.44	-	-	-	-	**50.95±0.07**
	20	39.61±0.53	41.36±0.20	-	-	-	-	**50.40±0.27**
	25	37.51±0.17	39.36±0.51	-	-	-	-	**50.02±0.20**

Table 2: Node classification performance (Accuracy±Std) under attack. We use metattack for regular-size graphs and PR-BCD for Arxiv. ”-” means not applicable. Performances of all baselines in Cora ML are not available and are run by ourselves. Other baseline performances are from Pro-GNN Jin et al. (2020). Bold symbols and underlines mean the first and second best performances respectively.

Ptb Rate (%)	GCN	GSL-LH	PPRGO	SoftMedian+PPRGO	GSL-LH+PPRGO
5	53.43 ± 0.27	54.56 ± 0.49	**58.18 ± 0.23**	57.24 ± 0.16	57.96 ± 0.48
10	46.75 ± 0.47	50.12 ± 0.67	53.39 ± 0.24	55.39 ± 0.29	**56.37 ± 0.47**
15	43.39 ± 0.62	50.95 ± 0.07	51.27 ± 0.17	54.56 ± 0.31	**54.83 ± 0.43**
20	40.17 ± 0.63	50.40 ± 0.27	50.31 ± 0.32	**54.40 ± 0.12**	53.98 ± 0.42
25	37.77 ± 0.65	50.02 ± 0.24	48.58 ± 0.40	**54.59 ± 0.26**	53.34 ± 0.40

Table 3: Node classification performance (Accuracy±Std) on ogbn-Arxiv under PR-BCD of different methods based on PPRGO.

adjacency matrix learned by GSL-LH can improve performances of PPRGO under attack. When attack rate is small ($5\%, 10\%, 15\%$), GSL-LH + PPRGO can outperform SoftMedian + PPRGO. However when attack rate is large ($20\%, 25\%$), GSL-LH + PPRGO's performances are slightly lower than SoftMedian + PPRGO.

4.4. Ablations

We mainly do ablation studies on ogbn-Arxiv to show the scalability of GSL-LH. There are two main parts we are interested in. In the first part, how good is lottery sampling versus traditional random sampling or feature-based selection when K-order graph is not applicable. The second part is how the sparsity of graphs and networks effects the performance of GSL-LH at different attack rate, which is important on the learned graph. To validate graph structure learning in large graphs, we visualize the learned graph structures.

Large graph sampling methods. In Table 4, we show the results of GSL-LH with different sampling methods at input graphs at different attack rates. Feature information is important when perturbation rate is high but can cause high variances under certain rates. Randomness is somehow good for sampling. Combining merits of both sampling methods, lottery sampling has achieved the best performance.

Prune sparsity. We evaluate different adjacent sparsity s_{adj} and weight sparsity s_w's performance on ogbn-Arxiv under different attack rates. Results are listed in Table 5 and 6 for perturbation rate of 0.05 and 0.25. Results of all rates are shown in Figure 3. In Table 5, we show that when the attack rate is not high, learned adjacency matrix scores are enough for good performance and do not need pruning. Higher pruning sparsity makes model more confused. When it comes to the situation that attack rate is high, as shown in Table 6, higher pruning sparsity in adjacency matrix will prompt the model to find better adjacency matrix resulting in the improvement of robustness. Besides, the high weight sparsity is harmful to models.

Ptb Rate (%)	Random	Feature	Lottery
5	52.84 ± 1.33	53.73 ± 0.63	**54.56 ± 0.49**
10	49.54 ± 0.71	49.20 ± 3.70	**50.12 ± 0.67**
15	50.70 ± 0.16	50.93 ± 0.29	**50.95 ± 0.07**
20	50.08 ± 0.19	45.27 ± 7.34	**50.40 ± 0.27**
25	49.69 ± 0.07	**50.22 ± 0.24**	50.02 ± 0.20

Table 4: Performances of different sampling methods in GSL-LH under different perturbation rates.

Acc. / Weight Sparsity / Adj Sparsity	None	0.2	0.4	0.6	0.8
None	**54.56 ± 0.49**	53.28 ± 1.53	52.73 ± 1.44	51.84 ± 0.85	49.76 ± 1.21
0.2	51.85 ± 0.52	50.89 ± 1.32	51.20 ± 0.73	51.02 ± 0.84	47.65 ± 1.15
0.4	48.90 ± 0.96	49.31 ± 0.52	47.45 ± 1.20	48.20 ± 1.16	47.02 ± 0.41
0.6	51.14 ± 0.17	49.80 ± 0.43	47.72 ± 1.44	43.39 ± 3.29	44.12 ± 3.78
0.8	52.34 ± 0.11	50.75 ± 0.62	48.60 ± 1.75	43.57 ± 4.59	11.00 ± 11.50

Table 5: Performances of different adjacent sparsity and weight sparsity under perturbation rate of 0.05. None means we don't prune the mask and maintain the trained scores in the model retrain step.

Visualization. To illustrate the explainability of GSL[4], we visualize the subgraph of original graph, attacked graph and purified graph by GSL-LH. In Figure 1, we show that

4. NetworkX (https://networkx.org) is used for our visualization.

Acc. / Weight Sparsity / Adj Sparsity	None	0.2	0.4	0.6	0.8
None	42.12 ± 1.67	41.57 ± 0.79	41.15 ± 0.45	42.12 ± 0.21	39.49 ± 1.25
0.2	41.33 ± 0.24	41.56 ± 0.19	40.17 ± 0.55	40.68 ± 0.18	25.14 ± 8.01
0.4	43.55 ± 0.21	43.57 ± 0.35	41.49 ± 1.07	38.79 ± 2.43	25.77 ± 9.12
0.6	46.36 ± 0.17	45.25 ± 0.41	43.51 ± 1.20	36.55 ± 5.23	26.12 ± 10.20
0.8	**50.02 ± 0.20**	48.55 ± 0.71	46.36 ± 1.72	41.37 ± 4.16	26.62 ± 11.33

Table 6: Performances of different adjacent sparsity and weight sparsity under perturbation rate of 0.25. None means we don't prune the mask and maintain the trained scores in the model retrain step.

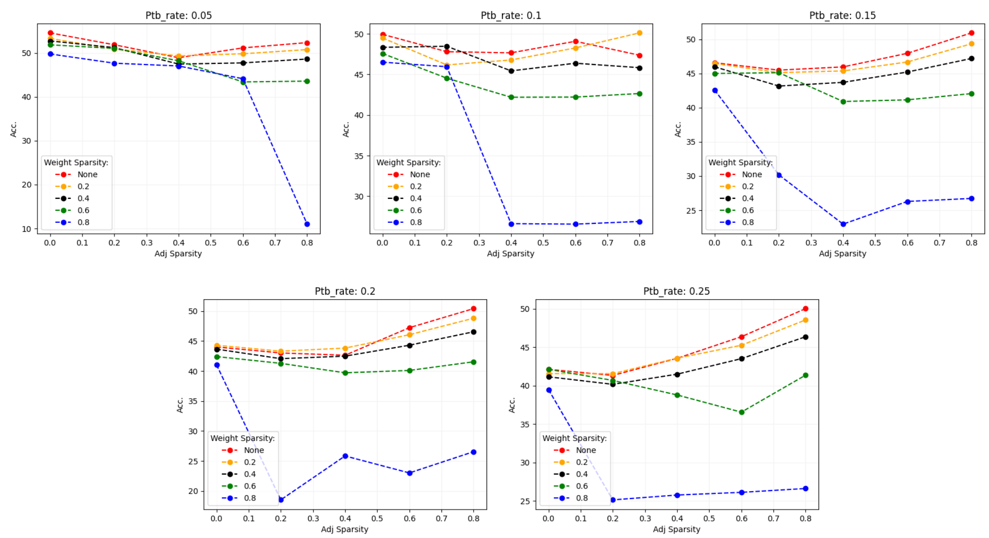

Figure 3: Performances of different adjacent sparsity under different weight sparsity with different perturbation rates. 0 in the x axis means None.

purified graph by GSL-LH can retain the deleted edges and delete redundant edges caused by attack, which could probably provide some insights into why GSL-LH is more robust than its backbone model.

5. Conclusion

In this paper, we propose the first scalable Graph Structure Learning method based on Lottery Hypothesis. Unlike previous graph structure learning methods which are highly dependent on dense representation of adjacency matrix, our GSL-LH can adopt sparse representation of adjacency matrix as input and easily be extended to large graphs, making GSL-LH the first graph structure learning method at scale. In experiments, GSL-LH performs better than its backbone model GCN without attack and shows state-of-the-art performances in regular-size graphs with metattack. In large graphs, with no previous GSL methods before, GSL-LH improves GCN's robustness against attack. While compared with defense methods other than GSL, GSL-LH achieves on par performance with Softmedian accompanied with PPRGO, which is the state-of-art defense method so far. Future works may include investigating other scalable GSL methods, and adding more large-scale graph datasets to generalize our algorithms.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (No.2022CSJGG0801), National Natural Science Foundation of China (No.62022027).

References

Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Graph neural networks with convolutional arma filters. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018.

Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2020. ACM.

Liang Chen, Jintang Li, Jiaying Peng, Tao Xie, Zengxu Cao, Kun Xu, Xiangnan He, and Zibin Zheng. A survey of adversarial learning on graphs. *arXiv preprint arXiv:2003.05730*, 2020a.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. *Advances in neural information processing systems*, 33:15834–15846, 2020b.

Tianlong Chen, Yu Cheng, Zhe Gan, Jingjing Liu, and Zhangyang Wang. Ultra-data-efficient gan training: Drawing a lottery ticket first, then training it toughly. *arXiv preprint arXiv:2103.00397*, 2021a.

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, 2021b.

Tianlong Chen, Zhenyu Zhang, Sijia Liu, Shiyu Chang, and Zhangyang Wang. Long live the lottery: The existence of winning tickets in lifelong learning. In *International Conference on Learning Representations*, 2021c.

Xiaohan Chen, Yu Cheng, Shuohang Wang, Zhe Gan, Zhangyang Wang, and Jingjing Liu. Earlybert: Efficient bert training via early-bird lottery tickets. *arXiv preprint arXiv:2101.00063*, 2020c.

Xuxi Chen, Zhenyu Zhang, Yongduo Sui, and Tianlong Chen. {GAN}s can play lottery tickets too. In *International Conference on Learning Representations*, 2021d.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 257–266, 2019.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271, 2020.

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.

Tyler Derr, Yao Ma, and Jiliang Tang. Signed graph convolutional networks. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 929–934. IEEE, 2018.

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.

Utku Evci, Fabian Pedregosa, Aidan Gomez, and Erich Elsen. The difficulty of training sparse neural networks. *arXiv*, abs/1906.10732, 2019.

Fuli Feng, Xiangnan He, Jie Tang, and Tat-Seng Chua. Graph adversarial training: Dynamically regularizing based on graph structure. *IEEE Transactions on Knowledge and Data Engineering*, 33(6):2493–2504, 2019.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. *arXiv*, abs/1912.05671, 2019a.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611*, 2019b.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv*, abs/1902.09574, 2019.

Zhe Gan, Yen-Chun Chen, Linjie Li, Tianlong Chen, Yu Cheng, Shuohang Wang, and Jingjing Liu. Playing lottery tickets with vision and language. *arXiv preprint arXiv:2104.11832*, 2021.

Xiang Gao, Wei Hu, and Zongming Guo. Exploring structure-adaptive graph learning for robust semi-supervised classification. In *2020 ieee international conference on multimedia and expo (icme)*, pages 1–6. IEEE, 2020.

Simon Geisler, Tobias Schmidt, Hakan cSirin, Daniel Zugner, Aleksandar Bojchevski, and Stephan Gunnemann. Robustness of graph neural networks at scale. In *NeurIPS*, 2021.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

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

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.

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.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR, 2018.

Neha Mukund Kalibhat, Yogesh Balaji, and Soheil Feizi. Winning lottery tickets in deep generative models, 2020.

Andrea Stevens Karnyoto, Chengjie Sun, Bingquan Liu, and Xiaolong Wang. Augmentation and heterogeneous graph neural network for aaai2021-covid-19 fake news detection. *International journal of machine learning and cybernetics*, pages 1–11, 2022.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

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

Juanhui Li, Harry Shomer, Jiayuan Ding, Yiqi Wang, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. Are message passing neural networks really helpful for knowledge graph completion? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10696–10711, July 2023.

Yawei Li, He Chen, Zhaopeng Cui, Radu Timofte, Marc Pollefeys, Gregory S Chirikjian, and Luc Van Gool. Towards efficient graph convolutional networks for point cloud handling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3752–3762, 2021a.

Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. Deeprobust: a platform for adversarial attacks and defenses. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35 (18):16078–16080, May 2021b.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *7th International Conference on Learning Representations*, 2019.

Haoyu Ma, Tianlong Chen, Ting-Kuei Hu, Chenyu You, Xiaohui Xie, and Zhangyang Wang. Good students play big lottery better. *arXiv preprint arXiv:2101.03255*, 2021a.

Yao Ma, Suhang Wang, Tyler Derr, Lingfei Wu, and Jiliang Tang. Graph adversarial attack via rewiring. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1161–1169, 2021b.

Zheng Ma, Junyu Xuan, Yu Guang Wang, Ming Li, and Pietro Liò. Path integral based convolution and pooling for graph neural networks. *Advances in Neural Information Processing Systems*, 33:16421–16433, 2020.

Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. In *8th International Conference on Learning Representations*, 2020.

Pedro Savarese, Hugo Silva, and Michael Maire. Winning the lottery with continuous sparsification. In *Advances in Neural Information Processing Systems 33 pre-proceedings*, 2020.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

Hannes Stärk, Octavian Ganea, Lagnajit Pattanaik, Regina Barzilay, and Tommi Jaakkola. Equibind: Geometric deep learning for drug binding structure prediction. In *International Conference on Machine Learning*, pages 20503–20521. PMLR, 2022.

Ke Sun, Zhouchen Lin, Hantao Guo, and Zhanxing Zhu. Virtual adversarial training on graph convolutional networks in node classification. In *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, pages 431–443. Springer, 2019.

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

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.

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

Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. *arXiv preprint arXiv:1906.04214*, 2019a.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019b.

Liang Yang, Zesheng Kang, Xiaochun Cao, Di Jin, Bo Yang, and Yuanfang Guo. Topology optimization based graph convolutional network. In *IJCAI*, pages 4054–4061, 2019.

Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Richard G. Baraniuk, Zhangyang Wang, and Yingyan Lin. Drawing early-bird tickets: Toward more efficient training of deep networks. In *8th International Conference on Learning Representations*, 2020.

Haonan Yu, Sergey Edunov, Yuandong Tian, and Ari S. Morcos. Playing the lottery with rewards and multiple languages: lottery tickets in rl and nlp. In *8th International Conference on Learning Representations*, 2020.

Dinghuai Zhang, Kartik Ahuja, Yilun Xu, Yisen Wang, and Aaron Courville. Can subnetwork structure be the key to out-of-distribution generalization? In *International Conference on Machine Learning*, pages 12356–12367. PMLR, 2021.

Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Yuanqi Du, Jieyu Zhang, Qiang Liu, Carl Yang, and Shu Wu. A survey on graph structure learning: Progress and opportunities. *arXiv e-prints*, pages arXiv–2103, 2021.

Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. *arXiv preprint arXiv:1902.08412*, 2019.

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.