WINNING INTACTNESS AND EFFICIENCY IN GRAPH LEARNING: A DUAL LOTTERY TICKET PERSPECTIVE

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Neural Networks (GNNs) have been widely recognized by virtue of achieving astonishing success in graph-related tasks. However, the computational overheads and memory footprint of fitting GNNs to large-scale graphs grow rapidly, which commonly poses obstacles to GNNs framework from being scaled up to physical large-scale graph applications. Graph Lottery Ticket (GLT) hypothesis firstly reveals that the existence of joint sparsity graph and network with admirable performance (called graph lottery tickets), and provide a novel insight for speeding up GNN-based model reasoning. From another perspective, the graph sparsification in GLT can be intepreted as a variant of sampling-based pooling, this however may result in loss of massive information and valid edges. In this work, addressing the above problems caused by sampling-based models, we combine gradually increasing regularization (GIR) and graph pooling tools to extrude information from discarded part to its complementary counterpart, *achieving a triple-win situation for sparsity, performance, and self-defined structure of co-graph and network.* Further, we rigorously prove our model can pick out noise and maintain reliable information in substructures in graph information bottleneck perspective. Extensive experimental results on various graph-related tasks validate the effectiveness of our algorithm.

1 INTRODUCTION

Graph Neural Networks (GNNs) Kipf & Welling (2016); Veličković et al. (2017); Hamilton et al. (2017) have emerged as the dominant model for a diversity of graph learning tasks, such as node classification Rong et al. (2019); Abu-El-Haija et al. (2020), link predictionZhang & Chen (2019); Hajiramezanali et al. (2019), and graph classification Ying et al. (2018); Gao & Ji (2019). The success of GNNs mainly derives from a recursive neighborhood aggregation scheme, i.e., message passing, in which each node updates its feature by aggregating and transforming the features of its neighbors. However, GNNs suffer notoriously high computational overheads when scaling up to large graphs or with dense connections, since conducting message passing over large or dense graphs proves costly for model training and inference Xu et al. (2018).

To alleviate such inefficiency, existing approaches mostly fall into two research lines – they either *simplify the graph structure* or *compressing the model*. Within the first class, many studies Chen et al. (2018); Eden et al. (2018); Calandriello et al. (2018) have investigated the use of sampling to reduce the computational footprint of GNNs. These sampling-based strategies are usually integrated with mini-batch training schedule for local feature aggregation and updating. Another representative is graph sparsification techniques Voudigari et al. (2016); Zheng et al. (2020); Li et al. (2020b) which improve training or inference efficiency of GNNs by learning to remove redundant edges from input graphs. In contrast to simplifying the graph structure, there are much fewer prior studies on pruning or compressing GNNs Tailor et al. (2020), as GNNs are generally less parameterized than DNNs in other fields, e.g., computer vision.

Surprisingly, Graph Lottery Ticket Hypothesis (GLT) Chen et al. (2021) has killed two birds with one stone, i.e., it has presented the first attempt to simultaneously simplify the input graph and prune the model for GNN inference. The key insight is to generalize the popular Lottery Ticket Hypothesis (LTH) Frankle & Carbin (2018) to GNNs. Recall that LTH articulates there always exist sparse and high-performance subnetworks in a dense network with random initialization (like winning tick-

ets in a lottery pool), GLT delineates a Graph Lottery Ticket as a combination of subgraph and sparse subnetwork with admirable performance. More specifically, GLT first devises a unified GNN sparsification (UGS) strategy for jointly pruning the graph adjacency matrix as well as the network weights, and then iteratively apply UGS to uncover the winning tickets in GNNs. Extensive experiments on GNN benchmarks have verified the effectiveness of GLT across various architectures, learning tasks, and initialization ways.

In essence, the graph simplification method in GLT is analogous to sampling-based graph pooling Ying et al. (2018); Wu et al. (2022); Roy et al. (2021) which selects important nodes for reducing graph representations while ideally preserving core structural information. Though promising, this sampling-based graph simplification in GLT may lead to two devastating challenges: (1) **Information loss:** Simply pruning sub-graph edges or network parameters may cause massive information loss, resulting in performance degradation. (2) **Aggregation failure:** As the sparsity increases, some "unimportant" edges may be eliminated by means of a pruning algorithm, but sometimes they connect two very important local communities (seen Fig. 1).

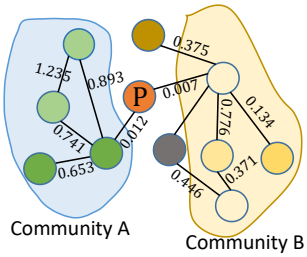

Figure 1: The blue and yellow background areas represent the community A and B, respectively. Various circles represent nodes, lines represent edges, and values represent the corresponding element values in the mask obtained by training. Node P has a very low connection weight, but connects two important areas.

In this work, we are committed to addressing the above-mentioned issues in sub-networks and sub-graphs acquisition. Our target is to transform a triple-win situation among self-defined structure, accuracy, and sparsity in joint sub-graphs and sub-networks. To meet the above challenges, we adopt GIR on self-defined structure to stimulate magnitude discrepancy among the weights. GIR can stimulate magnitude discrepancy among the weights, as regularization steady increase, the weights are naturally separated, and discarded weights are pushed to the position close to zero (regard as noise) Wang et al. (2020a). By this way, serviceable information is extruded into self-defined structure (reduce information loss). Meanwhile, due to the transformation of a substructure, the *aggregation failure* problem of the sampling-based method is avoided. In summarize, we divide whole process into three stages: (1) **Pre-train:** Different from sampling based pooling algorithms, we merge original graph local communities to formulate a small-size graph representation, which can remedy information loss issue throughout the pooling process. In this process, we use GNN layer to obtain assignment matrix for feature- or topology-based node assignments in each layer. Meanwhile, we add trainable matrices to element-wise product with output adjacent matrices. Finally, we add GIR on self-defined element in trainable matrices and weights for information extrusion. (2) **Pruning:** After pre-train, we prune the discarding weights in the network and elements in matrices, which are gradually suppressed via increasing regularization. (3) **Fine-tune:** we run pruned model on the graph data and adjust the values of the remaining weights. For the whole DGLT framework, we further explore the reliability from graph information bottleneck (GIB) perspective Wu et al. (2020). We show that our GIB design can overcome sampling-based designs in terms of avoiding information loss. Our work also can be linked with Dual Lottery Ticket Hypothesis (DLTH) Bai et al. (2022), so we named our algorithms as Graph Dual Lottery Ticket (DGLT) algorithm, and defined the obtained a sparse and admirable the pair of sub-nework and graph as *graph dual lottery tickets*. we make major contributions in this paper as follows:

- We present Graph Dual Lottery Ticket **(DGLT)** algorithm, which adopt *gradually increasing regularization* and *graph pooling* tools not only to realize the triple-win of self-defined structure, sparsity, accuracy of co-graph and network, but also can defeat the information loss and information aggregation failure issues of Graph Lottery Ticket (**GLT**) framework.
- Despite the simplicity and effectiveness, DGLT overcome the dual problem of GLT and generalize the idea of DLTH into GNN, whcih can significantly speeding up the inference of graph-related tasks. Notably that our DGLT algorithm can seamlessly apply to arbitrary form of GNNs in an plug-and-play fashion. All these advantages provide a feasible solution for future large-scale graph-based applications.
- We conduct extensive experiments to verify DGLT framework. For sparsity, For performance

2 RELATED WORK

Lottery Ticket Hypothesis (LTH): LTH demonstrates that *a highly sparse and independently trainable subnetwork can be identified from a dense model, by iterative pruning Frankle & Carbin (2018). The subnetwork can reach test accuracy comparable to the original network, which is called winning ticket.* LTH is initially observed in dense network and is broadly found in many fields currently, such as natural language processing (NLP), generative models and reinforcement learning Morcos et al. (2019); Evci et al. (2020); Frankle et al. (2020); Malach et al. (2020). Meanwhile, some derivative theories, such as Early Bird Lottery Tickets (EBT), are proposed to optimize the procedure of network sparseness and pruning Chen et al. (2020); You et al. (2021); Chen et al. (2021). In addition to the above work, Dual Lottery Ticket Hypothesis (DLTH) firstly considers a more general case to find the relationship between a dense network and its sparse counterparts Bai et al. (2022) – *it argues that a randomly selected subnetwork from a dense network can be transformed into a trainable condition and achieve admirable performance compared with the winning lottery pool, which suggests a more adjustable way of investigating sparse neural networks*. Our target is to find DLTH property in GNNs.

Graph Pooling: The idea of graph pooling stems from Convolutional Neural Networks (CNNs) LeCun et al. (2012) and pooling strategies can reduce the sizes of feature maps and enlarge receptive fields, thereby giving rise to better generalization and performance Ronneberger et al. (2015); Oktay et al. (2018); Zhou et al. (2018). Analogously, graph pooling operations have been adopted to amplify the topology receptive fields and reduce the input graph sizes. We divide graph pooling algorithms into two main categories. (1) Sampling-based graph pooling: this category focus on training a guiding projection vector to adaptively select a subset of nodes and then carry out graph reconstruction to form a small-size graph Gao & Ji (2019); Lee et al. (2019); Ranjan et al. (2020); Zhang et al. (2021). Though efficient, this node-drop design would inevitably result in massive information loss and isolated subgraphs, which will deteriorate the performance of GNNs. (2) Clustering-based pooling: this research line tries to merge original node communities and construct a new bunch of clusters Ying et al. (2018); Wu et al. (2022); Roy et al. (2021). This stream of graph pooling strategies can remedy the information loss problem of the sampling-based pooling methods.

Fast Graph Learning: Many graph-based sampling or compression algorithms are proposed to help large-scaled graph learning tasks. Despite graph pooling, Voudigari et al. (2016); Chen et al. (2018); Zeng et al. (2019); Ying et al. (2019; 2018) proposed different graph sampling algorithm to find the most representative sub-graph. Rong et al. (2019); Li et al. (2020b) adopted edge selection algorithm to remove redundant element in adjacency matrix to help fast graph learning. Faber et al. (2021) showed that there a significant amount of edges can be removed without harming the performance. GLTChen et al. (2021) firstly proposed the joint sparsification of graph-data and network, which generalize the LTH into GNN. Distinct from these sparsification GNN algorithms, we consider a more generalized case that transforms the co-graph and network into a suitable condition. We use the clustering-based GNN algorithm combined with GIR for information extrusion, this combination can easily overcome massive information loss and aggregation failure issues, thus having better applicability and practicability in the physical scenario.

3 METHODOLOGY

3.1 PRELIMINARIES

Graph Neural Network: Let $\mathcal{G} = \left\{A^{(0)}, X^{(0)}\right\}$ represent an input undirected graph-structured data with $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges. Input adjacency matrix $A = A^{(0)} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is defined to describe the overall graph topology, where $A\left[i,j\right] = 1$ if $(v_i, v_j) \in \mathcal{E}$ else $A\left[i,j\right] = 0$. Input feature $X = X^{(0)} \in \mathbb{R}^{|\mathcal{V}|\times F}$ denote the node feature matrix of the whole graph, where $x_i = X[i,:]$ is F-dimensional feature representation of node $v_i \in \mathcal{V}$. In particular, GNNs usually require A and X as inputs and realize partition information aggregation with specific "message-passing" function:

$$H^{(k)} = M\left(A, H^{(k-1)}; \Theta^{(k)}\right) \tag{1}$$

Where $H^{(k)} \in \mathbb{R}^{|\mathcal{V}|\times F}$ are node embedding after k-steps of GNN hierarchical aggregation. M is message propagation function and $\Theta^{(k)}$ are trainable parameters at GNN k-th layer. The input node

embedding $H^{(0)} = X$. Due to the popularity of graph neural networks, there are multitudinous propagation function M Gilmer et al. (2017); Hamilton et al. (2017) and variants of GNNs. For instance, a two-layer Graph Convolutional Networks (GCNs) Kipf & Welling (2016) can be written as:

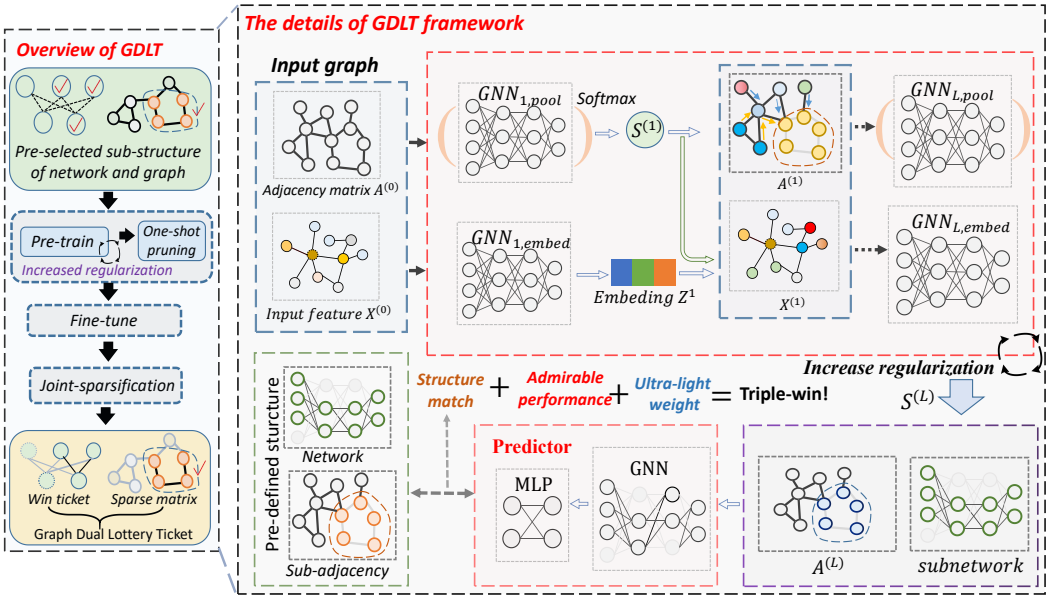

Figure 2: ***(Left)***The overview of our proposed DGLTH algorithm. ***(Right)***The details of DGLTH algorithm.

$$\underbrace{\mathcal{Z} = \mathcal{S}\left(\hat{A}\sigma\left(\hat{A}X\Theta^{(0)}\right)\Theta^{(1)}\right)}_{\text{predict function}} \quad \Leftrightarrow \quad \underbrace{\mathcal{L}\left(\mathcal{G},\Theta\right) = -\sum\nolimits_{v_i \in train\{\mathcal{V}\}} \mathrm{y}_i \log\left(\mathcal{Z}_i\right)}_{\text{loss function}} \tag{2}$$

Where $\mathcal{Z}$ is the prediction of GCN and $\mathcal{Z}_i$ represent predict value of node v_i, y_i are the ground-truth of node v_i and $\Theta = \left\{\Theta^{(0)}, \Theta^{(1)}\right\}$ are the weights of the whole network. $\sigma\left(\cdot\right)$ denote activation function. $\hat{A} = \tilde{D}^{-\frac{1}{2}}\left(A+I\right)\tilde{D}^{-\frac{1}{2}}$ is adjacency matrix with self-loops and $\tilde{D}_{ii} = \sum_j \hat{A}_{ij}$. In the whole GCN, we implement optimization process by cross entropy loss (Eq. 2 Right).

Graph Information Bottleneck (GIB): Information bottleneck (IB) originates from information transmission theory, which attempts to find a compression code of the input signal but retains as much valid information as possible from the original encoding Tishby et al. (2000). In recent years, with the rapid development of deep learning and its usage in various computer vision, natural language processing (NLP), and graph-related tasks, the information bottleneck theory is naturally transferred to deep networks and shows excellent effects Peng et al. (2018); Luo et al. (2019); Wang et al. (2020b); Wu et al. (2020); Yu et al. (2020); Miao et al. (2022). Our work builds upon graph field, as defined above, $Y = \left\{\mathrm{y}_1, \mathrm{y}_2 \ldots . \mathrm{y}_{|\mathcal{V}|}\right\}$ denotes the label of all nodes and $\mathcal{G}$ is the input graph. GIB-based methods try to find an optimal sub-graph $\mathcal{G}_s^*$ in sub-graph set $\mathbb{G}_{sub}\left(\mathcal{G}\right)$ by optimizing:

$$max_{\mathcal{G}_{sub} \in \mathbb{G}_{sub}(\mathcal{G})} I\left(\mathcal{G}_{sub}, Y\right) - \beta I\left(\mathcal{G}_{sub}, \mathcal{G}\right) \longrightarrow \mathcal{G}_s^* \tag{3}$$

$\mathcal{G}_{sub}$ denotes a sub-graph of the $\mathcal{G}$ and $I\left(\cdot\right)$ represents Shannon mutual information. β is the hyper-parameter used to control the proportion of the two parts. Compared with the GIB research line, we try to transform full graph into $\hat{\mathcal{G}}_s^*$ by a transformation function $T\left(*\right)$ (i.e., gradually increasing regularization) and satisfies Eq. 3, which considers a more counter-intuitive and general case that transform a dense network to its sparse dual winning tickets in GNNs.

3.2 Assignment Matrix for Sparse Sub-graph

As analogue of DIFFPOOL Ying et al. (2018) framework, DGLT pooling process adopt two parallel GNNs to learn embedding representation $Z^{(l)}$ and assignment matrix $s^{(l)}$ at layer l. For better

guidance subgraph generation, we impose a differentiable mask $m_A^{(l)}$ for $A^{(l)}$ in the hierarchical pooling process:

$$Z^{(l)} = GNN_{l,embed}\left(A^{(l-1)}, X^{(l-1)}; \Theta_{emb}^{(l)}\right) \tag{4}$$

$$s^{(l)} = softmax\left(GNN_{l,pool}\left(A^{(l-1)} \odot m_A^{(l-1)}, X^{(l-1)}; \Theta_{pool}^{(l)}\right)\right) \tag{5}$$

Where $\odot$ is the element-wise product. $GNN_{l,embed}$ and $GNN_{l,pool}$ are both standard GNN-framework, their aggregation function M make no assumption to any GNN structure, for which we can use any form of GNN as embedding and pooling tools. $s^{(l)} \in \mathbb{R}^{n_{l-1} \times n_l}$ $(n_{l-1} > n_l)$ in Eq. 5, in our implementation, we hope that n_{l-1} is slightly larger than n_l to ensure that graph information in the hierarchical clustering process extrudes steadily into small sub-structure. Given the embedding $Z^{(l)}$ and assignment matrix $s^{(l)}$, we apply the following equation to obtain adjacency matrix $A^{(l)}$ and embedding $X^{(l)}$ in the next layer:

$$A^{(l)} = {s^{(l)}}^T A^{(l-1)} s^{(l)}, \quad X^{(l)} = {s^{(l)}}^T Z^{(l)}. \tag{6}$$

In the last layer L, we can obtain a resilient sub-graph representation $\mathcal{G}_{sub} = \{A^{(L)}, X^{(L)}\}$.

3.3 Information extrusion for graph dual tickets

One of the most popular and simplest kinds of parameter penalty: the L_2 regularization commonly knowns as **weight decay**. L_2 regularization strategy draws the weights closer to the origin by adding a constraint term $\Omega(w) = \frac{1}{2}||w||_2^2$ to the loss function. DLTH Bai et al. (2022) employ a cleverly-used L_2 regularization to extrude information from unimportant part to complementary and pre-defined structure. DLTH demonstrates that when the regularization increases gradually, due to the different local curvature, the weights responses are distinguished and the larger curvature (corresponding to the larger Hessian matrix value) will move less. The difference between the weights will be separated by the cumulative regularization force LeCun et al. (1989); Wang et al. (2020a) (we have placed the proof in Appendix A). In our framework, the whole network (L-layers) parameter $\Theta = \left\{\Theta_{emb}^{(1)} \dots \Theta_{emb}^{(L)}, \Theta_{pool}^{(1)} \dots \Theta_{pool}^{(L)}, \Theta_{mlp}, \Theta_g\right\}$, the sets of subgraph representations of each layer $A = \left\{A^{(0)}, A^{(1)} \dots A^{(L)}\right\}$ and $X = \left\{X^{(0)}, X^{(1)} \dots X^{(L)}\right\}$. Trainable mask $m_A = \left\{m_A^{(0)}, m_A^{(1)} \dots m_A^{(L)}\right\}$. In the last layer of pooling, we directly use $m_A^{(L)} \odot A^{(L)}$ and send $\mathcal{G}_{sub}^{'} = \left\{m_A^{(L)} \odot A^{(L)}, X^{(L)}\right\}$ to GNN and MLP predictors for label forecasting. In total framework, DGLT can be achieved as optimization following objective function:

$$\mathcal{L}_{DGLT} := \mathcal{L}\left(\{m_A \odot A, X\}, \Theta\right) + \xi\,||m_A^*||_2^2 + \rho\,||\Theta^*||_2^2 \tag{7}$$

$$\xi^{(p+1)} = \begin{cases} \xi^{(p)} + \xi_a & \xi^{(p)} < \xi_{ceil} \\ \xi^{(p)} & \xi^{(p)} = \xi_{ceil} \end{cases} \qquad \rho^{(p+1)} = \begin{cases} \rho^{(p)} + \rho_a & \rho^{(p)} < \rho_{ceil} \\ \rho^{(p)} & \rho^{(p)} = \rho_{ceil} \end{cases} \tag{8}$$

As shown in Eq. 7, in addition to the cross-entropy loss, the loss function also contains two L_2 regularization terms, where m_A^* and Θ^* are non-important parameters predefined in matrices m_A and Θ, respectively. Finally, under the interplay of gradually increasing the regular force, the non-important weights are pushed to a position close to zero, and the information is extruded to the 'important' weights. In Eq. 8, $\xi^{(p)}$ and $\rho^{(p)}$ are the regularization terms at p-th updating. ξ_a and ρ_a indicate steps for increasing value. ξ_{ceil} and ρ_{ceil} indicate ceiling values of two regularization terms. We set range of regularization terms and control regularization value increases in a linear fashion until reaches their ceiling.

3.3.1 GIB view of gradually increasing regularization

GIB theory tries to identify unserviceable or noisy nodes of the training graph-structured data describe spurious correlation-versus-causations and enlarge the mutual information between remain part and ground-truth. Intuitively, a spurious correlation means that *after the introduction of nodes or graphs in the training set cannot increase or even reduce the mutual information between the*

training set and the label Glymour et al. (2016); Arjovsky et al. (2019); Krueger et al. (2021). In Eq. 3, The first term is used to maximize the mutual information of subgraphs and labels, and the second term wants the subgraph to be as small as possible. Different from sampling-based methods, DGLT algorithm considers a more general case to employ a transformation function to achieve information extrusion and guarantee spurious correlation removal. We will provide on theoretical analysis of how information extrusion can obtain a better performance sub-graph than sampling-based GIB algorithms.

Lemma 1: *when the penalty is increased at the same pace, because of different local curvature structures, the weights actually respond differently – weights with larger curvature will be less moved. As such, the magnitude discrepancy among weights will be magnified as regularization grows. Ultimately, the weights will naturally separate (unimportant weights tend to be very small and can be regarded as noise)* Wang et al. (2020a).

Based on **Lemma 1**, we list our two observations about spurious correlations distribution in graph: (1) Some nodes are pure noise; (2) Some node are composed of bootless correlations and useful associations. Under these two **Obs**ervations. We can get such solution: *Suppose each G contains subset* $\mathbb{G}_{sub}(\mathcal{G})$*, there exist* $\mathcal{G}_s^* \in \mathbb{G}_{sub}(\mathcal{G})$ *can remove spurious correlations, i.e.,* $I(\mathcal{G}_s^*, Y) \geq I(\mathcal{G}, Y)$*. Unserviceable or noisy information are distributed in graph node features. In our implementation, we can transform a* $\mathcal{G}$ *to* $\hat{\mathcal{G}}_s^*$ *and make sure that* $I\left(\hat{\mathcal{G}}_s^*, Y\right) \geq I(\mathcal{G}_s^*, Y) \geq I(\mathcal{G}, Y)$.

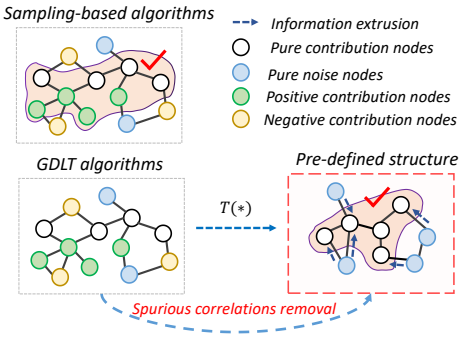

Figure 3: The white/blue node respectively indicate that all features can increase/decrease $I(\mathcal{G}, Y)$, while the green/yellow nodes indicate that there are useful associations and false associations in the nodes, but in the end, the useful/spurious association dominates

Theoretical Analysis. Obs 1: There exist pure noise nodes (note nodes set as $\dot{\mathcal{G}}_{sub}$) which make no contributions to $I(\mathcal{G}, Y)$ at all. Considering that our target is to maximize $I(\mathcal{G}_{sub}, Y)$:

$$I(\mathcal{G}_{sub}, Y) = I(Y; \mathcal{G}, \mathcal{G}_{sub}) - I(Y; \mathcal{G}|\mathcal{G}_{sub}) = I(Y; \mathcal{G}) - I(Y; \mathcal{G}|\mathcal{G}_{sub}) \tag{9}$$

Where the first equality is because *Chain Rule for Mutual Information* and the third equality is because $\mathcal{G}_{sub} \in \mathbb{G}_{sub}(\mathcal{G})$. $I(Y; \mathcal{G})$ is given from the beginning, so $I(Y; \mathcal{G})$ is a constant. Our target is to maximize $I(\mathcal{G}_{sub}, Y)$ (i.e., minimize the $I(Y; \mathcal{G}|\mathcal{G}_{sub})$). Since all nodes in $\dot{\mathcal{G}}_{sub}$ are noise, $I\left(\dot{\mathcal{G}}_{sub}, Y\right)$ in$I(\mathcal{G}, Y) - I\left(\mathcal{G} \backslash \dot{\mathcal{G}}_{sub}, Y\right) = I\left(\dot{\mathcal{G}}_{sub}, Y\right)$ reach the minimum. In GIB, $\mathcal{G}_s^* = \mathcal{G} \backslash \dot{\mathcal{G}}_{sub}$. For gradually increasing regularization, we can extrusion information to pre-defined structure. Since $\dot{\mathcal{G}}_{sub}$ are unserviceable, the amount of information squeezed is 0 in transformed $\hat{\mathcal{G}}_s^*$. Based on the above inference, we can obtain $I(\mathcal{G}_s^*, Y) = I\left(\hat{\mathcal{G}}_s^*, Y\right)$ in **Obs 1**.

Theoretical Analysis. Obs 2: The spurious information is distributed in some nodes and these nodes are composed of valid information and spurious information.

Suppose nodes set with spurious information is $\dot{\mathcal{G}}_{sub} = \left\{\dot{\mathcal{G}}_{sub}^*, \overline{\dot{\mathcal{G}}_{sub}^*}\right\}$, where nodes in $\dot{\mathcal{G}}_{sub}^*$ make positive contributions to $I(\mathcal{G}, Y)$ and $\overline{\dot{\mathcal{G}}_{sub}^*}$ makes no (even negative) contributions to $I(\mathcal{G}, Y)$. $I(\mathcal{G}, Y)$ reaches the maximum when $\mathcal{G}_s^* = \mathcal{G} \backslash \overline{\dot{\mathcal{G}}_{sub}^*}$. In our implementation, we extrude information to $\hat{\mathcal{G}}_s^*$ by training masks, and unimportant weights in masks tend to be exceedingly small and can be regarded as noises. In this circumstance, $\hat{\mathcal{G}}_s^*$ contains at least more useful information than $\hat{\mathcal{G}}_s^*$ under the same sparseness as $\hat{\mathcal{G}}_s^*$. We can obtain $I\left(\hat{\mathcal{G}}_s^*, Y\right) > I(\mathcal{G}_s^*, Y) > I(\mathcal{G}, Y)$ in **Obs 2**.

To summarize, DGLT can obtain a comparable (even better) sub-graph compare to the sampling-based GIB algorithm.

4 EXPERIMENTS

In this section, we conduct extensive experiments to answer following research questions: **RQ1:** How effective is our proposed DGLT algorithm in transforming sparse sub-graphs or sub-networks? **RQ2:** How scalable is DGLT algorithm on large-scale datasets? **RQ3:** Under a certain sparse ratio of subgraph or subnetwork, how does another term affect the model performance? **RQ4:** How should the mini-step values of the increased regularization force be set?

4.1 EXPERIMENTAL SETTINGS

Datasets. Six benchmarks for graph related task are selected in this paper to verify the effectiveness of our DGLT algorithm. In our implementation, we choose three popular and regular scale graph datasets Core, Citeseer (small-size) and PubMed (medium-size) Kipf & Welling (2016) for node classification and link prediction tasks. To observe scalability of DGLT, we further use large-scaled dataset Cgbl-Collab Hu et al. (2020) for link prediction task. Finally, we use D&D Dobson & Doig (2003) and ENZYMES Borgwardt et al. (2005) as benchmarks and test our algorithm in graph classification task. The statistics of these six datasets are reported in Table 1.

Table 1: ROC-AUC (Receiver Operating Characteristic-Area Under the Curve) value is equivalent to the probability that a randomly chosen positive example is ranked higher than a randomly chosen negative example. Hit@50 means that taking the candidate edge of the top 50, the proportion of the 50 edges is predicted correctly.

Dataset	tasks	Grpahs	nodes	edges	features	classes	Metrics
Cora	Node classification/Link prediction	–	2,708	5,429	1,443	7	Accuracy/ROC-AUC
Citeseer	Node classification/Link prediction	–	3,327	4,732	3,703	6	Accuracy/ROC-AUC
PubMed	Node classification/Link prediction	–	19,717	88,338	500	3	Accuracy/ROC-AUC
Ogbl-Collab	Link prediction	–	235,868	1,285,465	128	2	Hits@50
D&D	Graph classification	1,178	334,925	16,886,092	89	2	Accuracy
ENZYMES	Graph classification	600	19580	745654	3	6	Accuracy

Backbones and parameter setting. For all selected backbones and datasets, we compare our DGLT algorithm with GLT Chen et al. (2021) and random pruning algorithm under the same network settings. For regular-size graph datasets, we here adopt GCN Kipf & Welling (2016), GIN Xu et al. (2018) and GAT Veličković et al. (2017) as backbones. For large-scaled graph Cgbl-Collab, we use 28-layer deep ResGCNs Li et al. (2020a) for link prediction. In the last, to observe performance on graph classification task, we adopt GraphSAGE Hamilton et al. (2017) on D&D and ENZYMES datasets. More experimental setting details we have placed in Appendix B.

4.2 MAIN EXPERIMENTS (RQ1)

To answer **RQ1**, we fairly train and test our DGLT algorithm with other methods under the same training settings as described in Appendix C. We respectively control that adjacency matrix is unchanged [1] (the graph sparsity is zero) and the weight sparsity is zero, and observe the influence of gradual sparsity of one of the terms on the results. As displayed in Fig. 5, (additional experiments are placed in Appendix 7), we make the following observations: (1) **DGLT consistently outperforms GLT and random pruning under the same sparsity of graph and weights.** For three datasets, we randomly select the self-defined ratio of the graph or parameters, we find that compare with GLT and random pruning, DGLT can get high-quality dual lottery tickets. Specially, for PubMed+GIN combination, graph dual lottery is obtained with 85.0% graph sparsity and nearly 87% weight sparsity. For Citeseer+GIN, we can also get a graph dual lottery with nearly 65.0% graph sparsity and 91.0% weight sparsity. (2) **DGLT can obtain ultra-lightweight subnetworks.** For Cora and Citeseer small-scale datasets on node classification task, until 90% sparsity, the model still shows no significant performance degradation (even surpass the baselines performance). For relative large PubMed dataset, DGLT can even achieve 85.0% graph sparsity without performance degradation. We demonstrate that DGLT transform graph dual tickets by GIR, which can remedy the imformtion loss and get a informative pair of sub-graph and sub-network.(3) **The extreme graph sparsity of graph dual tickets depends on the property of graphs.** Though DGLT can help find

[1] Different from pruning, we transform graphs into small size representations and note $1 - \frac{||A^{(L)}||_0}{|\mathrm{A}|}$ as sparsity ratio. $||\cdot||_0$ and $|\cdot|_0$ are the number of non-zero elements and total number of elements, respectively.

extreme sparse graph dual tickets (e.g., PubMed+GIN or PubMed+GAT), this phenomenon does not occur in small data, which demonstrate that transforming dual winning tickets depends on property of graph. We consider that graph dual winning tickets may be easier to transform in large datasets and we will explore this question in depth in Sec. 4.3.

4.3 Scalability of DGLT on large-scale dataset (RQ2)

To answer **RQ2**, we use 28-layer ResGCNs on Ogbl-Collab dataset. We demonstrate the potential of DGLT for the real-world usage by evaluating the trade-off among the reasoning time overhead, accuracy, and memory savings. As shown in Fig.4, we can list two observations: (1) Similar to the main experiment, DGLT outperforms GLT and random pruning among different graph sparsity and weight sparsity. In implementation, we can get a dual lottery ticket with nearly 70% graph sparsity and 85% weight sparsity, where respectively outperforms about 6% and 7% of GLT under the same sparsity. (2) DGLT is more stable on large datasets. Similar to DGLT on PubMed dataset, as graph sparsity increased (until nearly 90% sparsity), the performance showed a slow decline, but there was a very obvious fluctuation in the small datasets, which shows that DGLT is more stable and conducive to expanding to large-scale datasets.

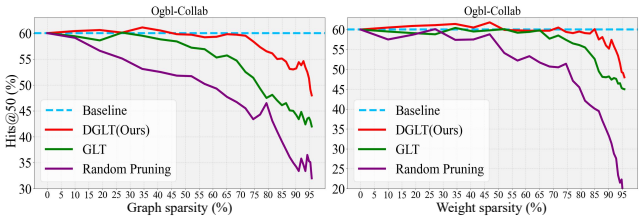

Figure 4: The results for link prediction task on Ogbl-Collab+ResGCNs settings.

4.4 Ablation studies on graph and weight sparsification ratio

We control graph sparsity P_g or weight sparsity P_θ at a fixed ratio and observe the performance of the model under different sparsity of another item. As shown in Fig. 6, we list the observation as follow: (1) Take an overall look, different lines are basically in similar positions. As we control P_θ at 10%, 30%, 50% and 70% values, we can find we can get dual lottery tickets with nearly 30% weight sparsity. Particularly, when we set $P_\theta = 70\%$, we can obtain admirable sub-graph representations only with about 2% performance degradation. As we control P_g at 10%, 30%, 50% and 70% values, the position of each line is more clearly distinguished. Notably that when $P_g = 30\%$, we can get an expressive subnetwork with 70% sparsity compare with original network. (2) As the P_g, P_θ continues to increase, model performance just shows a slightly degradation. These results experimentally show that our algorithm can help find an ultra-lightweight and small sub-graph representation, which can achieve comparable performance with original backbones. From the practical usage perspective, our ablation studies give a feasible solution for co-graphs and networks joint-sparsification (More ablation study experiments are placed in Appendix D).

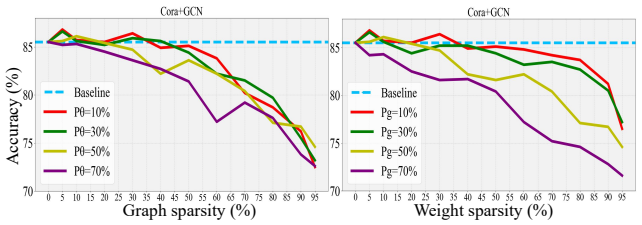

Figure 6: Ablation studies of pruning ratios for transforming sparse graph p_g and network p_θ on Cora+GCN setting.

4.5 Mini-step values of regularization

To answer **RQ4**, we control graph sparsity as 30%, weight sparsity as 70% and choose multiple mini-step values of regularization (ξ_a and ρ_a take the same mini-step values) for comparison. As shown in Tab. 2, we can make conclusion as follows: **Mini-step values of regularization should maintain in a small regime.**. In our implementation, we should control increased regularization steps on the order of $1e-6$. Meanwhile, when the progressive regularization force maintain in larger regime ($> 5e-4$), the model effect decreases significantly. We demonstrate that the process of information extrusion should be slow and excessive steps of increased regularization will cause the model to fall into a ill-conditioned solution.

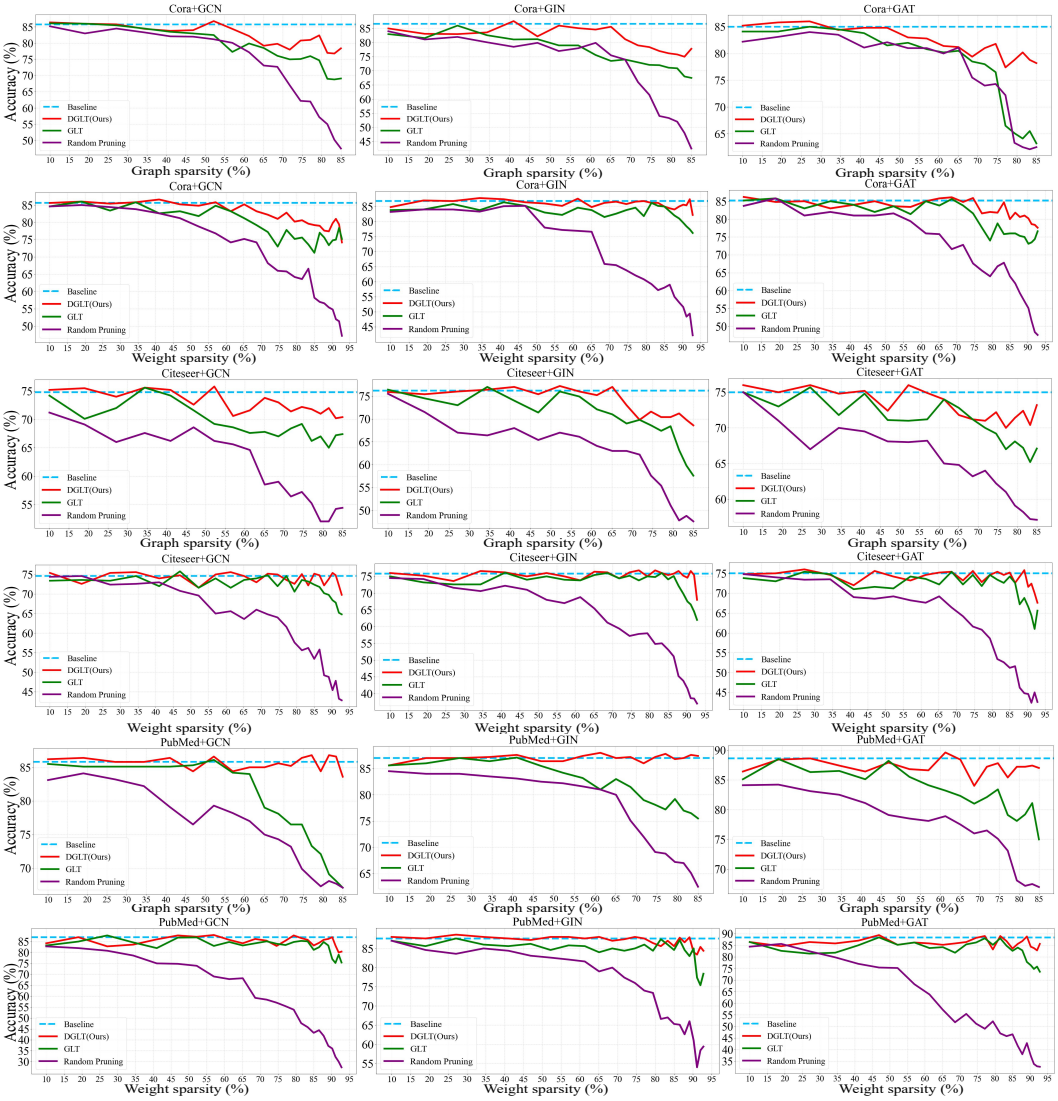

Figure 5: **node classification** performance over Cora, Citeseer and PubMed with GCN/GIN/GAT backbones, respectively. Blue dash lines represent the baseline performance of full GNNs on full graphs.

Table 2: Different mini-step values of regularization over Cora/Citeseer/PubMed with different backbones. For clarity, the highest/second-highest performances are emphasized with red/blue fonts.

Settings		Mini-steps of regularization						
Datasets	Methods	0	1e-6	5e-6	1e-5	5e-5	1e-4	5e-4
Cora	GCN (baseline=86.7%)	84.9%	86.7%	86.3%	85.1%	84.3%	82.3%	81.4%
	GIN (baseline=85.9%)	84.4%	85.9%	85.4%	84.7%	84.0%	82.7%	80.1%
	GAT (baseline=85.4%)	83.9%	85.4%	85.9%	85.1%	83.4%	81.1%	80.2%
Citeseer	GCN (baseline=75.7%)	74.9%	75.7%	76.2%	76.8%	75.4%	71.7%	71.9%
	GIN (baseline=76.3%)	74.5%	76.3%	76.0%	73.4%	74.4%	72.1%	71.1%
	GAT (baseline=76.6%)	75.3%	76.6%	75.8%	74.9%	74.8%	72.8%	70.9%
PubMed	GCN (baseline=85.0%)	84.3%	85.1%	85.7%	83.4%	83.1%	81.8%	79.8%
	GIN (baseline=84 .8%)	84.1%	84.8%	84.3%	83.1%	82.8%	81.4%	80.4%
	GAT (baseline=85.7%)	84.1%	85.7%	85.1%	83.5%	82.7%	81.0%	79.9%

5 Conclusion

In this paper, we adopt versatile and simple tools, i.e., regularization and graph pooling, to simultaneously address high computational overheads and large memory footprints issues. Though sampling-

based methods such as GLT have given feasible solutions for graph sparse learning, our DGLT can easily overcome information loss and aggregation failure problems in sampling-based algorithm by progressive information extrusion. Compared with GLT, our DGLT consider a more general case in graph which can help achieve triple-win among sparsity, performance, and self-defined structure, being summarized as follows: (1) GLT concentrates solely on sampling for a pair of sub-graphs and sub-networks by iterative magnitude pruning, which is not universal to practical usages and may result in massive information loss. (2) DGLT uses GIR to extrude information from discarded weight to self-defined structure, parallelly, we adopt GIR on pooling process to form a refined graph representation. Theoretical analysis of graph information bottleneck guarantees the reliability and resilient of our algorithm. We experimentally verify our DGLT algorithm among various backbones and datasets (including large-scale dataset) and results repeatedly prove the effectiveness of our idea.

REFERENCES

Sami Abu-El-Haija, Amol Kapoor, Bryan Perozzi, and Joonseok Lee. N-gcn: Multi-scale graph convolution for semi-supervised node classification. In *uncertainty in artificial intelligence*, pp. 841–851. PMLR, 2020.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Yue Bai, Huan Wang, Zhiqiang Tao, Kunpeng Li, and Yun Fu. Dual lottery ticket hypothesis. *arXiv preprint arXiv:2203.04248*, 2022.

Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005.

Daniele Calandriello, Alessandro Lazaric, Ioannis Koutis, and Michal Valko. Improved large-scale graph learning through ridge spectral sparsification. In *International Conference on Machine Learning*, pp. 688–697. PMLR, 2018.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

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*, pp. 1695–1706. PMLR, 2021.

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*, 2020.

Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

Talya Eden, Shweta Jain, Ali Pinar, Dana Ron, and C Seshadhri. Provable and practical approximations for the degree distribution using sublinear graph samples. In *Proceedings of the 2018 World Wide Web Conference*, pp. 449–458, 2018.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pp. 2943–2952. PMLR, 2020.

Lukas Faber, Yifan Lu, and Roger Wattenhofer. Should graph neural networks use features, edges, or both? *arXiv preprint arXiv:2103.06857*, 2021.

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

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pp. 3259–3269. PMLR, 2020.

Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pp. 2083–2092. PMLR, 2019.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Madelyn Glymour, Judea Pearl, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, 2016.

Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. Variational graph recurrent neural networks. *Advances in neural information processing systems*, 32, 2019.

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.

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

David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pp. 5815–5826. PMLR, 2021.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International conference on machine learning*, pp. 3734–3743. PMLR, 2019.

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

Jiayu Li, Tianyun Zhang, Hao Tian, Shengmin Jin, Makan Fardad, and Reza Zafarani. Sgcn: A graph sparsifier based on graph convolutional networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 275–287. Springer, 2020b.

Yawei Luo, Ping Liu, Tao Guan, Junqing Yu, and Yi Yang. Significance-aware information bottleneck for domain adaptive semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6778–6787, 2019.

Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pp. 6682–6691. PMLR, 2020.

Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In *International Conference on Machine Learning*, pp. 15524–15543. PMLR, 2022.

Ari Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. *Advances in neural information processing systems*, 32, 2019.

Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018.

Xue Bin Peng, Angjoo Kanazawa, Sam Toyer, Pieter Abbeel, and Sergey Levine. Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow. *arXiv preprint arXiv:1810.00821*, 2018.

Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5470–5477, 2020.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Kashob Kumar Roy, Amit Roy, AKM Mahbubur Rahman, M Ashraful Amin, and Amin Ahsan Ali. Structure-aware hierarchical graph pooling using information bottleneck. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

Shyam A Tailor, Javier Fernandez-Marques, and Nicholas D Lane. Degree-quant: Quantization-aware training for graph neural networks. *arXiv preprint arXiv:2008.05000*, 2020.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

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

Elli Voudigari, Nikos Salamanos, Theodore Papageorgiou, and Emmanuel J Yannakoudakis. Rank degree: An efficient algorithm for graph sampling. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 120–129. IEEE, 2016.

Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. Neural pruning via growing regularization. *arXiv preprint arXiv:2012.09243*, 2020a.

Rundong Wang, Xu He, Runsheng Yu, Wei Qiu, Bo An, and Zinovi Rabinovich. Learning efficient multi-agent communication: An information bottleneck approach. In *International Conference on Machine Learning*, pp. 9908–9918. PMLR, 2020b.

Junran Wu, Xueyuan Chen, Ke Xu, and Shangzhe Li. Structural entropy guided graph hierarchical pooling. In *International Conference on Machine Learning*, pp. 24017–24030. PMLR, 2022.

Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. Graph information bottleneck. *Advances in Neural Information Processing Systems*, 33:20437–20448, 2020.

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

Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnn explainer: A tool for post-hoc explanation of graph neural networks. *arXiv preprint arXiv:1903.03894*, 2019.

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.

Haoran You, Zhihan Lu, Zijian Zhou, and Yingyan Lin. Gebt: Drawing early-bird tickets in graph convolutional network training. *arXiv preprint arXiv:2103.00794*, 2021.

Junchi Yu, Tingyang Xu, Yu Rong, Yatao Bian, Junzhou Huang, and Ran He. Graph information bottleneck for subgraph recognition. *arXiv preprint arXiv:2010.05563*, 2020.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graph-saint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

Muhan Zhang and Yixin Chen. Inductive matrix completion based on graph neural networks. *arXiv preprint arXiv:1904.12058*, 2019.

Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Zhao Li, Chengwei Yao, Dai Huifen, Zhi Yu, and Can Wang. Hierarchical multi-view graph pooling with structure learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*, pp. 11458–11468. PMLR, 2020.

Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pp. 3–11. Springer, 2018.

A PROOF OF GRADUALLY INCREASING REGULARIZATION

Regularization is long deemed as a tool on limiting the capacity of deep learning networks, by adding a penalty term $\Omega(\Theta)$ to the objective function J. We denote the regularized objective loss function by $\tilde{J}$:

$$\tilde{J}(\Theta; \mathrm{X}, \mathrm{y}) = J(\Theta; \mathrm{X}, \mathrm{y}) + \alpha\Omega(\Theta) \tag{10}$$

Where $\alpha \in [0, \infty)$ is a hyperparameter that weights the relative contribution of the penalty term. Setting α to 0 results in no regularization. Larger values of α correspond to more regularization. For L_2 regularization (commonly known as weight decay), $\Omega(\Theta) = \frac{1}{2}||w||_2^2$ is added into objective function. To simplify the presentation, we assume no bias parameter, so Θ is just equaled to w and we can list objective function as:

$$\tilde{J}(\Theta; \mathrm{X}, \mathrm{y}) = J(\Theta; \mathrm{X}, \mathrm{y}) + \frac{\alpha}{2} w^T w \tag{11}$$

Where the gradient of objective function is:

$$\nabla_w \tilde{J}(\mathrm{w}; \mathrm{X}, \mathrm{y}) = \alpha w + \nabla_w J(\mathrm{w}; \mathrm{X}, \mathrm{y}) \tag{12}$$

The whole network can be optimized by:

$$w \leftarrow w - \epsilon(\alpha w + \nabla_w J(w; X, y)) \tag{13}$$

Written another way, the update is:

$$w \leftarrow (1 - \epsilon\alpha)w - \nabla_w J(w; X, y) \tag{14}$$

Further we simplify the analysis by making a quadratic approximation to the objective function in the neighborhood of the weights that obtains minimal unregularized training cost, $w^* = argmin_w(w)$. In the neighborhood of w^*, we can obtain:

$$J(\Theta; \mathrm{X}, \mathrm{y}) = J(w^*) + \frac{1}{2}(w - w^*)^T H(w - w^*) \tag{15}$$

Where H is the Hessian matrix of J with respect to w evaluated at w^*. There is no first-order term in this quadratic approximation, because w^* is defined to be a minimum, where the gradient vanishes. Meanwhile, due to the w^* is the minimum of J, we can conclude that H is positive-semidefinite. The minimum of $\tilde{J}$ occurs where its gradient:

$$\nabla_w \widetilde{J}(w) = H(w - w^*) = 0 \tag{16}$$

Gradually increasing regularization (GIR): We add weight decay gradient in 16, as regularized $\tilde{J}$ reaches minimum (corresponding to $\tilde{w}$), we can obtain:

$$\alpha\tilde{w} + H\left(\tilde{w} - w^*\right) = 0 \quad \Longrightarrow \quad \tilde{w} = \left(H + \alpha I\right)^{-1} H w^* \tag{17}$$

After increasing the penalty α by $\delta\alpha$, the new converged weights $\hat{w}$ have the same relation (17) with last convergence point $\bar{w}$:

$$\hat{w} = \left(H + \delta\alpha I\right)^{-1} H \bar{w} \tag{18}$$

Where I represents identity matrix. As analysis in Wang et al. (2020a), we list two simplified cases to move forward. (1) H is diagonal. For $\hat{w}_i$ with second derivative h_{ii}. As adding $\delta\alpha > 0$ regularized force, the new converged weights can be proved to be

$$\hat{w}_i = \frac{h_{ii}}{h_{ii} + \delta\alpha}\bar{w}_i \quad \Longrightarrow \quad \frac{\hat{w}_i}{\bar{w}_i} = \frac{1}{\delta\alpha / h_{ii} + 1} \tag{19}$$

Where $\frac{\hat{w}_i}{\bar{w}_i} \in [0, 1)$ since $h_{ii} \geq 0$ and $\delta\alpha > 0$. As seen, when $h_{ii} \uparrow$, make $\frac{\hat{w}_i}{\bar{w}_i} \uparrow$, we can find that the weight relatively less moves towards the origin.

(2) We consider a general case (2-d) instead of diagonal matrix of H. $\bar{w} = \begin{Bmatrix} \bar{w}_1 \\ \bar{w}_2 \end{Bmatrix}$, $H = \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix}$, $\hat{H} = \begin{pmatrix} h_{11} + \delta\alpha & h_{12} \\ h_{21} & h_{22} + \delta\alpha \end{pmatrix}$. After adding mini-step value of regularization,

$$\begin{Bmatrix} \hat{w}_1 \\ \hat{w}_2 \end{Bmatrix} = \frac{1}{\left|\hat{H}\right|} \begin{Bmatrix} \left(h_{11}h_{22} + h_{11}\delta\alpha - h_{12}^2\right)\bar{w}_1 + \delta\alpha h_{12}\bar{w}_2 \\ \left(h_{11}h_{22} + h_{22}\delta\alpha - h_{12}^2\right)\bar{w}_2 + \delta\alpha h_{12}\bar{w}_1 \end{Bmatrix} \approx \frac{1}{\left|\hat{H}\right|} \begin{Bmatrix} \left(h_{11}h_{22} + h_{11}\delta\alpha - h_{12}^2\right)\bar{w}_1 \\ \left(h_{11}h_{22} + h_{22}\delta\alpha - h_{12}^2\right)\bar{w}_2 \end{Bmatrix} \tag{20}$$

$$\Longrightarrow \frac{\hat{w}_1}{\bar{w}_1} = \frac{1}{\left|\hat{H}\right|}\left(h_{11}h_{22} + h_{11}\delta\alpha - h_{12}^2\right) \quad \frac{\hat{w}_2}{\bar{w}_2} = \frac{1}{\left|\hat{H}\right|}\left(h_{11}h_{22} + h_{22}\delta\alpha - h_{12}^2\right) \tag{21}$$

As seen, we can draw the same conclusion: $h_{11} > h_{22}$ also results in $\frac{\hat{w}_1}{\bar{w}_1} > \frac{\hat{w}_2}{\bar{w}_2}$. To summarize, when the penalty is increased at the same pace, due to the different local curvature structure, the weights response in a different way. Weights with larger curvature will be less moved. As regularization gradually increases, the magnitude discrepancy among weights will be magnified and the unimportant weights tend to zero and can be regarded as noise.

B EXPERIMENTAL SETTINGS

Train-val-test Splitting of Datasets. To rigorously verify the effectiveness of our proposed DGLT algorithm, we control the network designs consistent under the same task. As for node classification task of regular-size datasets, we follow the same data split criteria as GLT Chen et al. (2021), i.e., 140 (Cora), 120 (Citeseer) and 60 (PubMed) labeled data for training, 500 nodes for validation and 1000 nodes for testing. As for link prediction, we shuffle the datasets and sample 80% edges for training, 10% for validation, 10% for testing, respectively. For Ogbl-Collab, in order to simulate a real collaborative recommendation application, we take the cooperation before 2017 as the training edge, the cooperation in 2018 as the validation edge and the cooperation in 2019 as the testing edge. For graph classification task, we choose D&D and ENZYMES datasets. D&D and ENZYMES includes graphs of protein structures, in which a node represents an amino acid. We perform 10-fold cross-validation to observe model performance and reported the accuracy averaged over 10-fold.

Backbone settings. As for regular-scale datasets Cora, Citeseer and PubMed, we adopt GCN/GIN/-GAT backbones for node classification and link prediction tasks. In our implementation, we adopt 3-layer GCN, 3-layer GIN and 3-layer GAT, respectively. For link prediction in Ogbl-Collab, we adopt 28-layer ResGCNs. For graph classification, we hope to obtain a small size graph representation and we use GraphSAGE for forecasting. Concretely, we select three pooling layers for graph size scaling. After the last pooling layer, we add a GrpahSAGE layer and MLP for prediction. Further, we place the training details and hyper-parameter configuration in Table 3.

Table 3: Training details and hyper-parameter configuration. $\xi^{(0)}$ and $\rho^{(0)}$ indicate the starting value of the graph regularization and weight regularization, respectively. ξ_a and ρ_a indicate the size of the graph regularization and weight regularization increase value.

Computing Infrastructures: three NIVIDIA Tesla v100 (16GB GPU) Software Framework: Pytorch									
Task	Node classification			Link prediction				Graph classification	
Dataset	Cora	Citeseer	PubMed	Cora	Citeseer	PubMed	Ogbl-collab	D&D	ENZYMES
Epochs (pre-train/fine-tune)	500/200	500/200	700/200	500/200	500/200	700/200	500/200	500/300	500/300
$\xi^{(0)}$	5e-04	5e-04	5e-04	5e-04	5e-04	5e-04	5e-04	5e-04	5e-04
ξ_a	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06
$\rho^{(0)}$	5e-04	5e-04	5e-04	5e-04	5e-04	5e-04	5e-04	5e-04	5e-04
ρ_a	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06	1e-06
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam
learning rate	0.008	0.01	0.01	0.008	0.01	0.01	0.01	0.0001	0.0001

C ADDITIONAL EXPERIMENTS TO ANSWER RQ1

More experiments on **link prediction task** are shown in Fig. 7. We make observations as follows: (1) DGLT aggressively improves the reasoning efficiency without significant performance degradation. For Cora dataset, we can get dual lottery tickets with nearly 50% graph sparsity and 80% weight sparsity. For Citeseer dataset, we can get dual lottery tickets with nearly 70% graph sparsity and 85% weight sparsity. For PubMed dataset, we can get dual lottery tickets with nearly 60% graph sparsity and 80% weight sparsity. (2) DGLT can transfer seamlessly to different graph-related tasks. In addition to the node classification task, GDLT model can also show better results on the link prediction task. (3) We can get similar conclusions as in the main experiments, DGLT can achieve better results than GLT and random pruning under the same sparsity ratio.

Further, we conduct 3-layer GraphSAGE Hamilton et al. (2017) counterpart with DGLT algorithm on ENEYMES and D&D datasets under different weight sparsity ratio. Similar to the original graph pooling method DIFFPOOL settings, we control the graph sparsity as 95% (total two pooling layers in DIFFPOOL) and observe the performance under the different weight sparsity. As shown in Table 4, it is not difficult to find that the information extrusion algorithm well maintains the accuracy under high sparsity. On ENEYMES dataset, We can observe that the expressiveness of the model could be improved even pruning 80% weights. Meanwhile, when the weight sparsity of the model reached 90%, the performance of the model still did not decrease significantly: it only decreased by 2.24% on the ENEYMES dataset and by 0.62% on the D&D dataset. In a nutshell, these results demonstrate that our DGLT algorithm can also be easily generalized to graph classification tasks, which further illustrates that DGLT has resilient pluggability.

Table 4: Graph classification accuracies in percent over ENZYMES and D&D datasets under different weight sparsity.

DIFFPOOL+ENZYMES; Baseline accuracy:63.21% Graph sparsity=95%											
Weight sparsity (%)	5	10	20	30	40	50	60	70	80	90	95
DGLT (Hits@50)	63.34	65.27	65.44	66.73	63.87	62.49	62.17	63.34	61.77	60.97	61.47
DIFFPOOL+D&D; Baseline accuracy:77.33% Graph sparsity=95%											
Weight sparsity (%)	5	10	20	30	40	50	60	70	80	90	95
DGLT (Hits@50)	78.36	80.15	77.48	76.39	77.81	78.50	77.51	77.36	78.39	76.71	75.15

D ADDITIONAL ABLATION RESULTS

In addition to Cora+GCN setting on Sec. 4.4, we summarize Citeseer+GAT experimental results on **link prediction task** and PubMed+GIN experimental results on **node classification task**. As shown in Fig. 8, we can further draw similar conclusions that our DGLT algorithm can obtain dual lottery tickets, which can both save memory footprints and speed up reasoning efficiency.

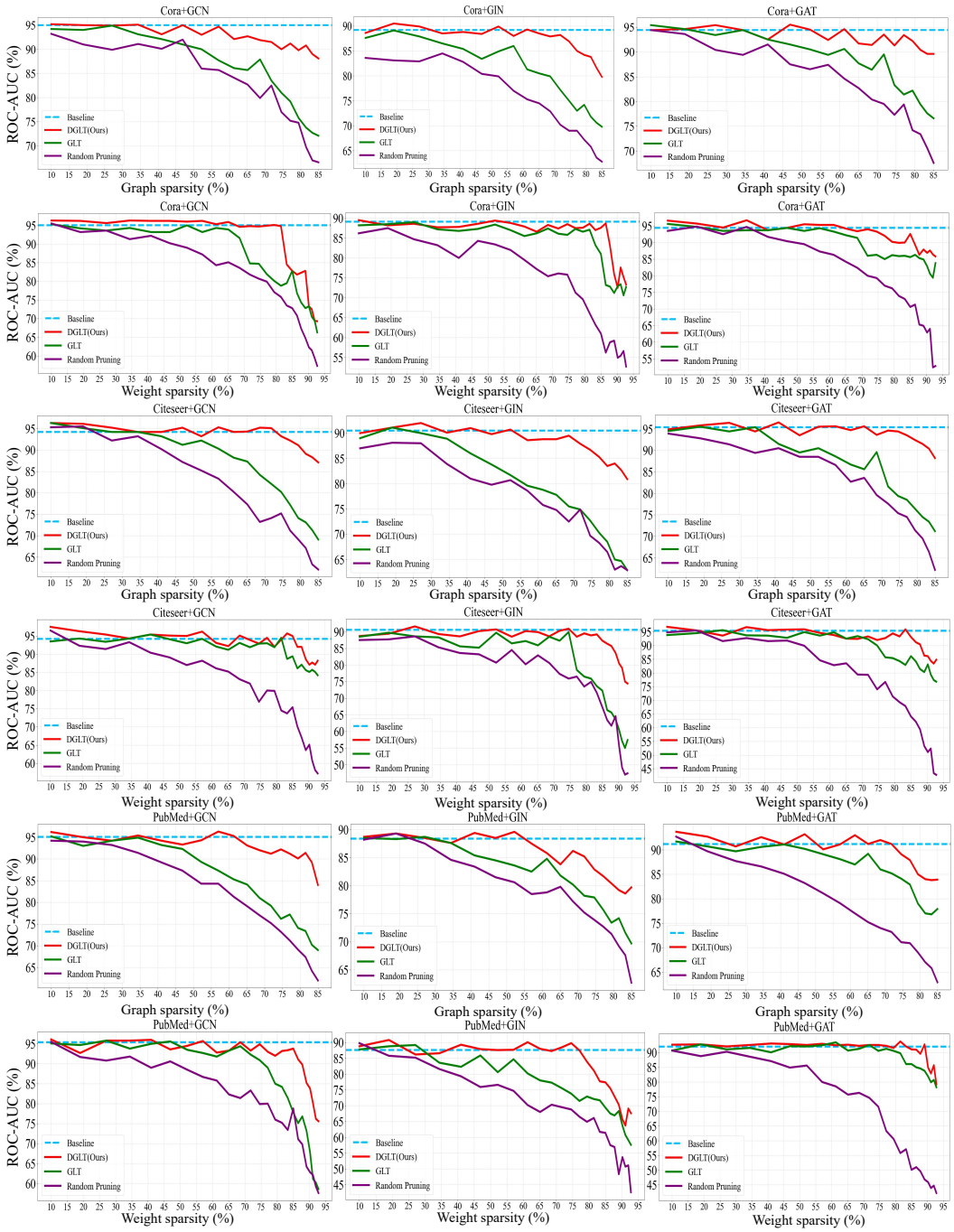

Figure 7: The results for additional **link prediction task**. We adopt Cora/Citeseer/PubMed as benchmarks and test our DGLT alogrithms on GCN/GIN/GAT three backbones for comparison. Blue dash lines represent the baseline performance of full GNNs on full graphs.

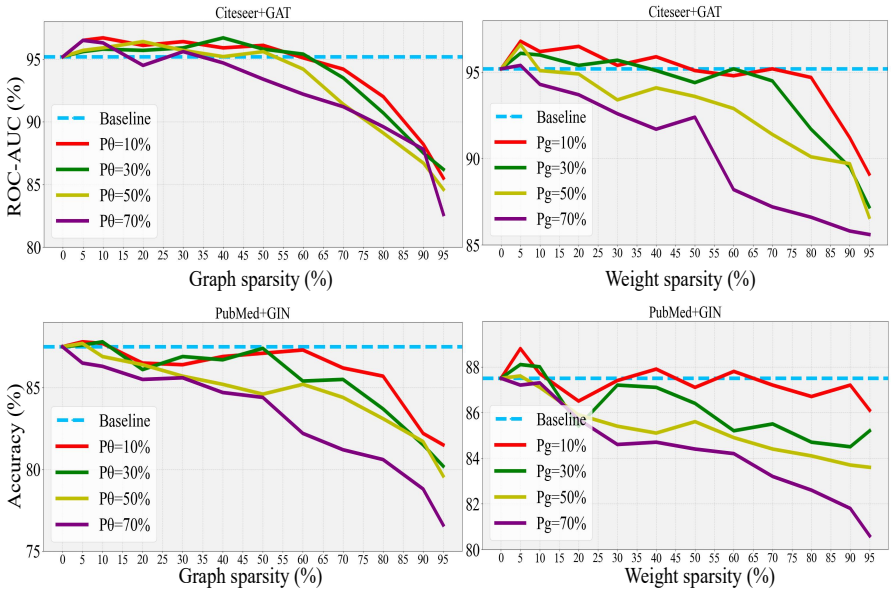

Figure 8: Ablation studies of pruning ratios for transforming sparse graph p_g and network p_θ, we select Citeseer+GAT for link prediction task and PubMed+GIN for node classification task. Bule dash lines represent model performance over full graph counterpart with full networks.