Sui YD, Wang X, Chen TL *et al.* Inductive Lottery Ticket Learning for Graph Neural Networks. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 33(1): 1–15 January 2018. DOI 10.1007/s11390-015-0000-0

Inductive Lottery Ticket Learning for Graph Neural Networks

Yong-Duo Sui[1] (隋勇铎), Xiang Wang[1,*] (王　翔), *Member, CCF,* Tianlong Chen[2] (陈天龙), Meng Wang[3] (汪萌), *Fellow, IEEE,* Xiang-Nan He[1,*] (何向南), *Member, CCF,* Tat-Seng Chua[4] (蔡达成)

[1] *School of Data Science, University of Science and Technology of China, Hefei 230027, China*

[2] *Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin 78712, U.S.A.*

[3] *School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China*

[4] *School of Computing, National University of Singapore, Singapore*

E-mail: syd2019@mail.ustc.edu.cn; xiangwang@ustc.edu.cn; tianlong.chen@utexas.edu; wangmeng@hfut.edu.cn; hexn@ustc.edu.cn; dcscts@nus.edu.sg

Received June 28, 2022; accepted October 9, 2023.

Abstract Graph neural networks (GNNs) have gained increasing popularity, while usually suffering from unaffordable computations for real-world large-scale applications. Hence, pruning GNNs is of great need but largely unexplored. The recent work UGS studies lottery ticket learning for GNNs, aiming to find a subset of model parameters and graph structure that can best maintain the GNN performance. However, it is tailed for the transductive setting, failing to generalize to unseen graphs, which are common in inductive tasks like graph classification. In this work, we propose a simple and effective learning paradigm, Inductive Co-Pruning of GNNs (ICPG), to endow graph lottery tickets with inductive pruning capacity. To prune the input graphs, we design a predictive model to generate importance scores for each edge based on the input; to prune the model parameters, it views the weight's magnitude as their importance scores. Then we design an iterative co-pruning strategy to trim the graph edges and GNN weights based on their importance scores. Although it might be strikingly simple, ICPG surpasses the existing pruning method and can be universally applicable in both inductive and transductive learning settings. On ten graph-classification and two node-classification benchmarks, ICPG achieves the same performance level with 14.26%-43.12% sparsity for graphs and 48.80%-91.41% sparsity for the model.

Keywords lottery ticket hypothesis, graph neural networks, neural network pruning

1 Introduction

Graph neural networks (GNNs) [1–3] have become a prevalent solution for machine learning tasks on graph-structured data. Such success is usually ascribed to the powerful representation learning of GNN, which incorporates the graph structure into the representations, such as aggregating neural messages from the neighboring nodes to update the ego node's representation.

As the field grows, there is an increasing need of building deeper GNN architectures [4,5] on larger-scale graphs [6]. While deepening GNNs shows potential on large-scale graphs, it also brings expensive computations due to the increased scale of graph data and model parameters, limiting their deployment in resource-constrained applications. Taking fraud detection in a transaction network as an example, the scale of user nodes easily reaches millions or even larger, making a GNN-detector model prohibitive to stack deep layers and predict malicious behaviors in real-time. Hence, pruning over-parameterized GNNs is of great need, which aims to answer the question: Can we co-sparsify the input graphs and model, while preserving

Regular Paper

This work was supported by the National Key Research and Development Program of China under Grant No. 2020YFB1406703, and the National Natural Science Foundation of China under Grant No. 9227010114.

[*] Corresponding Author

or even improving the performance?

Recently, a pruning approach, UGS [7], has been proposed to find graph lottery tickets (GLTs) — smaller subsets of model parameters and input graphs. At its core is Lottery Ticket Hypothesis (LTH) [8] speculating that any dense, randomly-initialized neural network contains a sparse subnetwork, which can be trained independently to achieve a matching performance as the dense network. Specifically, UGS employs trainable masks on each edge in the input graph and each weight in the model parameters, to specify their importance. When training the model with the masks, the strategy of iterative magnitude-based pruning (IMP) [8] is used to discard the edges and weights with the lowest mask values at each iteration.

Despite the effectiveness, there exist the following limitations: (1) UGS focuses solely on providing transductive graph masks by generating a painstakingly customized mask for a single edge individually and independently. That is, the edge masks are limited to the given graph, making UGS infeasible to be applied in the inductive setting since the edge masks hardly generalize to unseen edges or entirely new graphs. (2) Applying a mask for each edge alone only provides a local understanding of the edge, rather than the global view of the entire graph (e.g., in node classification) or multiple graphs (e.g., in graph classification). Moreover, the way of creating trainable edge masks will double the parameters of GNNs, which violates the purpose of pruning somehow. As a result, these edge masks could be suboptimal to guide the pruning. (3) The unsatisfactory graph pruning will negatively influence the pruning of model weights. Worse still, low-quality weight pruning will amplify the misleading signal of edge masks in turn. They influence each other and form a vicious circle. We ascribe all these limitations of UGS to its transductive nature. Hence, conducting combinatorial pruning in the inductive setting is crucial to high-quality winning tickets.

In this work, we emphasize the inductive nature within the combinatorial pruning of input graphs and GNN parameters and present our framework, Inductive Co-Pruning of GNNs (ICPG). It is an extremely simple but effective pruning framework that is applicable to any GNN in both inductive and transductive settings. Specifically, for the input graphs, we design a predictive model, AutoMasker, which learns to generate edge masks from the observed graphs. It is parameterized with an additional GNN-based encoder, whose parameters are shared across the population of observed graphs. As a consequence, AutoMasker is naturally capable to specify the significance of each edge and extract core subgraphs from a global view of the entire observations. For the model parameters, we simply exploit the magnitude of a model weight to assess whether it should be pruned, rather than training an additional mask. Having established the edge masks and weight magnitudes, we can obtain high-quality GLTs by pruning the lowest-mask edges and lowest-magnitude weights. Experiments on ten graph classification and two node classification datasets consistently validate our framework ICPG by identifying high-quality GLTs. Moreover, we inspect the GNN-level and graph-level transferability, which promises for deploying ICPG in the pre-training and fine-tuning paradigm to save the computational cost. The visualizations show that ICPG always retains decisive subgraphs, such as edges located on digital pixels in MNIST graphs, which further illustrates rationality and explainability.

In all, our main contributions can be summarized as follows.

- We introduce ICPG, an innovative pruning framework, capable of pruning both the GNN model and input graphs, which excels at identifying high-quality GLTs across diverse graph representation tasks in both inductive and transductive settings.
- We have validated ICPG's capacity to find GLTs in datasets of various scales through extensive experiments, while maintaining performance with a range of graph sparsity from 22.62% to 43.12% and GNN sparsity from 67.23% to 91.41%.
- We demonstrate that ICPG offers transferability at both the GNN and graph levels, resulting in im-

proved performance and lower computational costs in downstream tasks. This is substantiated by thorough comparisons, analyses, and visual inspections that validate its effectiveness, applicability, and explainability.

2 Related Work

Graph Neural Networks (GNNs) [1–3, 9, 10] have emerged as a powerful tool for learning the representation of graph-structured data. The great success mainly comes from the structure-aware learning, which follows the iterative message-passing scheme. Specifically, we denote an undirected graph by $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ with the node set $\mathcal{V}$ and edge set $\mathcal{E}$. $\boldsymbol{A} \in \{0, 1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the adjacency matrix, where $A[i, j] = 1$ denotes the edge between node v_i and node v_j, otherwise $A[i, j] = 0$. $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$ is the matrix of node features, where $\boldsymbol{x}_i = \boldsymbol{X}[i, :]$ is the d-dimensional feature of the node $v_i \in \mathcal{V}$. Given a K-layer GNN, its k-th layer generates the representation of node v_i as in (1) and (2).

$$\boldsymbol{a}_i^{(k)} = AGGREGATION^{(k)}(\{\boldsymbol{h}_j^{(k-1)}|j \in \mathcal{N}(i)\}), \quad (1)$$

$$\boldsymbol{h}_i^{(k)} = COMBINE^{(k)}(\boldsymbol{h}_i^{(k-1)}, \boldsymbol{a}_i^{(k)}), \quad (2)$$

where $\boldsymbol{h}_i^{(k)}$ and $\boldsymbol{a}_i^{(k)}$ are the representation of node v_i and the message aggregated from its neighbor nodes set $\mathcal{N}(i)$, respectively; the *AGGREGATION* and *COMBINE* operators are the message passing and update functions, respectively. After propagating through K layers, we get the final representations of nodes, which facilitate downstream node-level tasks, such as node classification and link prediction. As for graph-level tasks like graph classification and graph matching, we further hire the *READOUT* function to generate the representation of the whole graph $\mathcal{G}$, as in (3).

$$\boldsymbol{Z}_{\mathcal{G}} = READOUT(\{\boldsymbol{h}_i^{(k)}|v_i \in \mathcal{V}, k \in \{1, ..., K\}\}). \quad (3)$$

Various GNNs [2, 9, 10] adopt different *AGGREGATION*, *COMBINE* and *READOUT* functions to refine the desired information.

Lottery Ticket Hypothesis (LTH) [8] states that a sparse subnetwork exists in a dense randomly-initialized network that can be trained to achieve comparable performance to the full models. LTH is explored in many fields such as computer vision and natural language processing [11–15]. Recently, UGS [7] extends the LTH to the GNNs, proposing the Graph Lottery Ticket (GLT), which includes subgraph and subnetwork pairs that can be trained independently to reach comparable performance to the dense pairs. However, due to the transductive nature of graph-specific masks, UGS [7] cannot develop in inductive learning settings. To address this issue, we have incorporated AutoMasker into our approach. This tool possesses the capability to learn the importance of each edge from training graphs on a global scale and predict significance scores for newly introduced graphs. By being both graph-agnostic and inductive, AutoMasker effectively surmounts the limitations traditionally associated with graph-specific masks, thus paving the way for novel advancements within inductive learning settings.

Table 1. Comprehensive Comparisons in the Inference Stage

Method	Sparse Graph		Sparse Model
	Transductive	Inductive	
SGAT [16]	✓	✗	✗
NeuralSparse [17]	✓	✓	✗
GraphSAGE [21]	✓	✓	✗
DropEdge [18]	✓	✓	✗
SGCN [19]	✓	✗	✗
GEBT [22]	✓	✗	✓
UGS [7]	✓	✗	✓
ICPG (Ours)	✓	✓	✓

Graph Sparsification and Sampling aim to find core subgraphs in graph learning. Numerous strategies [16–27] are proposed to achieve efficient training or inference. SGAT [16] adopts sparse attention to remove edges. NeuralSparse [17] utilizes a DNN to identify task-irrelevant edges. Sampling-based methods [18–21] sample and aggregate features from a node's local neighborhood. DropEdge [18] randomly drops edges from the input graph, which can be seen as a data augmenter. Another research line selects subgraphs in an optimization way. SGCN [19] and GEBT [22] adopt the ADMM optimization algorithm to sparsify the adjacency matrix. UGS [7] utilizes trainable masks to prune graphs. Unfortunately, these methods either fail to utilize sparse graphs in the inductive inference stage or do not use sparse GNNs for efficient inference. Distinct from them, ICPG endows GNNs with inductive sparsification capacity, which can universally work in both transductive and inductive settings with both sparse graphs and models. We make comprehensive comparisons with the above methods in Table 1.

3 Preliminaries

In this section, we first briefly introduce the inductive graph learning. Then we formulate the task of learning graph lottery tickets under inductive setting.

3.1 Inductive Graph Learning

Before entering our method, we first clarify the inductive learning settings of our work. Compared with inductive graph learning, transductive graph learning denotes that unlabeled test data can be used in the training process. For example, in semi-supervised node classification tasks [1], training and test nodes form an entire graph. During model training, we need to take the full graph data as input and predict the class of test nodes based on all node features (including test node features), all edges, and labels of training nodes. Hence, all information (except labels) on the test data is available during training. In contrast, inductive graph learning means that all information of the test graph is not available during the training process. For example, in graph classification tasks, we use the training data at hand to train GNN models, hoping that the models can effectively generalize to the unseen test data. Compared with the transductive graph learning setting, inductive graph learning cannot utilize any information from test data. Therefore, inductive learning requires better generalization ability of the model. Unfortunately, UGS [7] designs learnable weights for all edges of an entire graph. This training strategy requires that the topological structure of the graph data is fixed and invariant between the training and inference stage. Hence, UGS [7] is only possible to apply to the setting of transductive graph learning, while it cannot apply to inductive graph learning.

3.2 Inductive Graph Lottery Ticket (GLT)

Without loss of generality, we consider the task of graph classification as an example. Given a GNN classifier $f(\cdot, \boldsymbol{\Theta}_{g_0})$, it starts from the randomly-initialized parameters $\boldsymbol{\Theta}_{g_0}$ before training and arrives at the well-optimized parameters $\boldsymbol{\Theta}_g$ after training. Once trained, it takes any graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ as the input and yields a probability distribution over C classes $\hat{\boldsymbol{y}} = f(\mathcal{G}, \boldsymbol{\Theta}_g)$.

Learning GLTs aims to make the input graph $\mathcal{G}$ and the model weights $\boldsymbol{\Theta}_{g_0}$ sparse to reduce the computational costs, while preserving the performance. Formally, it aims to generate two masks $\boldsymbol{m}_{\mathcal{G}}$ and $\boldsymbol{m}_{\Theta}$, which are applied on $\mathcal{G}$ and $\boldsymbol{\Theta}_{g_0}$ correspondingly, so as to establish the sparser input graph $\mathcal{G}' = (\boldsymbol{m}_{\mathcal{G}} \odot \boldsymbol{A}, \boldsymbol{X})$ and initialized weights $\boldsymbol{\Theta}'_{g_0} = \boldsymbol{m}_{\Theta} \odot \boldsymbol{\Theta}_{g_0}$. Hereafter, through retraining the subnetwork $f(\cdot, \boldsymbol{\Theta}'_{g_0})$ on the sparse versions $\{\mathcal{G}'\}$ of training graphs, we can get the new converged parameters $\boldsymbol{\Theta}'_g$. If the well-optimized subnetwork can achieve comparable performance with full graphs and network, we term the pair of $\mathcal{G}'$ and $f(\cdot, \boldsymbol{\Theta}'_{g_0})$ as a GLT. Although a recent study, UGS [1], proposes to learn the GLTs, it focuses solely on the transductive setting but leaves the inductive setting untouched. Specifically, it assigns a trainable mask to each edge of the input graph and trains such graph-specific masks individually and independently. As a consequence, these edge-dependent masks are limited to the given graph, hardly generalizing to unseen edges or entirely new graphs. Distinct from UGS, we aim to uncover GLTs in the inductive learning setting.

4 Methodology

In this section, we propose a novel pruning framework, named Inductive Co-Pruning of GNNs (ICPG), to find the GLTs. We first introduce the key component in ICPG, named AutoMasker. Then we present our inductive strategy of co-pruning the input graphs and model parameters.

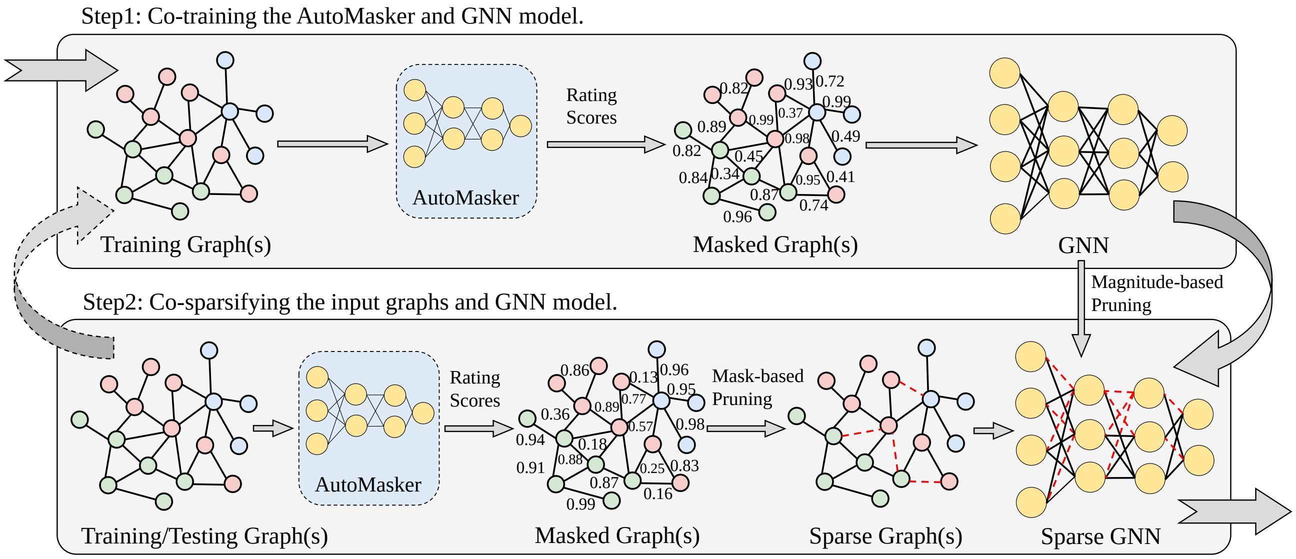


Fig.1. ICPG framework to find the GLTs.

4.1 AutoMasker

Instead of assigning a mask to a single edge, our idea is extremely simple: we take a collection of graph instances and design a trainable model to learn to mask edges collectively. The key ingredient of this model is an additional GNN-based model, termed AutoMasker, whose parameters are shared across the population of observed graphs. Here we represent AutoMasker as the combination of a graph encoder and a subsequent scoring function. Formally, given a graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$, AutoMasker applies a GNN-based graph encoder $g(\cdot)$ to create representations of all nodes as:

$$\boldsymbol{H} = g(\boldsymbol{A}, \boldsymbol{X}),$$

where $\boldsymbol{H} \in \mathbb{R}^{|\mathcal{V} \times d|}$ stores d-dimension representations of all nodes, whose i-th row $\boldsymbol{h}_i$ denotes the representation of node v_i; $g(\cdot)$ is a GNN following the message-passing paradigm in (1). To assess the importance score of edge (i, j) between node v_i and node v_j, AutoMasker builds a multi-layer perceptron (MLP) upon the concatenation of node representations $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$, which yields the score α_{ij}. In what follows, the sigmoid function $\sigma(\cdot)$ projects α_{ij} into the range of $(0, 1)$, which represents the probability of edge (i, j) being the winning ticket. The scoring function is represented as follows:

$$s_{ij} = \sigma(\alpha_{ij}), \quad \alpha_{ij} = \text{MLP}([\boldsymbol{h}_i, \boldsymbol{h}_j]).$$

By employing the scoring function over all possible edges, we are able to collect the matrix of edge masks $\boldsymbol{s}_{\mathcal{G}}$, where $\boldsymbol{s}_{\mathcal{G}}[i, j] = s_{ij}$ if edge (i, j) holds, otherwise $\boldsymbol{s}_{\mathcal{G}}[i, j] = 0$. In a nutshell, we summarize the AutoMasker function as:

$$\boldsymbol{s}_{\mathcal{G}} = \text{AutoMasker}(\mathcal{G}, \boldsymbol{\Theta}_a), \tag{4}$$

where $\boldsymbol{\Theta}_a$ is the parameter of AutoMasker, covering the parameters of the GNN encoder and the MLP.

Although the key ingredient of AutoMasker is simple, it has several conceptual advantages over UGS.

Algorithm 1 Mask & Magnitude-based Pruning

```
Input: $\mathcal{D}$, $f(\cdot, \boldsymbol{\Theta}_{g_0})$, $AutoMasker(\cdot, \boldsymbol{\Theta}_{a_0})$, $\mathcal{M}$, $\boldsymbol{m}_{\Theta}$, Epoch $T$.
Output: Sparsified masks $\{\boldsymbol{m}'_{\mathcal{G}_i}\}_{i=1}^{N}$, $\boldsymbol{m}'_{\Theta}$.
for $t = 0$ to $T - 1$ do
    for $\mathcal{G}_i \in \mathcal{D}$ and $\boldsymbol{m}_{\mathcal{G}_i} \in \mathcal{M}$ do
        $\mathcal{G}_i \leftarrow (\boldsymbol{m}_{\mathcal{G}_i} \odot \boldsymbol{A}_i, \boldsymbol{X}_i)$
        $\boldsymbol{s}_{\mathcal{G}_i} \leftarrow AutoMasker(\mathcal{G}_i, \boldsymbol{\Theta}_{a_t})$
        $\mathcal{G}_i \leftarrow (\boldsymbol{s}_{\mathcal{G}_i} \odot \boldsymbol{A}_i, \boldsymbol{X}_i)$
        Forward $f(\mathcal{G}_i, \boldsymbol{m}_{\Theta} \odot \boldsymbol{\Theta}_{g_t})$
        Backward to update $\boldsymbol{\Theta}_{a_{t+1}}$, $\boldsymbol{\Theta}_{g_{t+1}}$
    end for
end for
for $\mathcal{G}_i \in \mathcal{D}$ do
    $\boldsymbol{s}_{\mathcal{G}_i} \leftarrow AutoMasker(\mathcal{G}_i, \boldsymbol{\Theta}_{a_T})$
    Set $p_g = 5\%$ of the lowest mask values in $\boldsymbol{s}_{\mathcal{G}_i}$ to
    0 and others to 1, creating $\boldsymbol{m}'_{\mathcal{G}_i}$.
    Prune $p_\theta = 20\%$ of the lowest magnitude parameters in $\boldsymbol{\Theta}_{g_T}$, creating $\boldsymbol{m}'_{\Theta}$.
end for
```

1) Global view. Although edge masks derived from UGS might preserve the fidelity to local importance, they do not help to delineate the general picture of the whole graph population. Distinct from UGS, AutoMasker takes a global view of making decisions. Specifically, for instance-level, AutoMasker adopts GNN as its backbone. Because of the message-passing mechanism of GNN, AutoMasker can fully consider the topology information of the entire graph data. For the dataset-level, all graph data share an AutoMasker, therefore it can make decisions for each edge in each graph data from a global perspective by observing all graph data in the dataset. As edges usually collaborate to make predictions, rather than working individually, they form a coalition like the functional groups of a molecule graph, the community of a social network. Hence, AutoMasker will learn the invariant and stable patterns in training data and can well transfer the learned patterns to the unseen test data.

2) Lightweight edge masks. When using UGS to prune graph data with millions of edges or nodes, the cost of assigning local edge masks one by one will be prohibitive with such a large-scale dataset in real-world scenarios. Moreover, UGS introduces additional parameters, whose scale remains the same as the edge number $\sum_{\mathcal{G}} |\mathcal{E}|$ and is much larger than the original parameters being pruned. Hence, it somehow violates the purpose of pruning. In our AutoMasker, the additional parameter is Θ_a in (4), which remains invariant across the change of data scale.

3) Generalization. In contrast to UGS, AutoMasker can generalize the mechanism of mask generation to new graphs without retraining, making it more efficient to prune unseen and large-scale graphs. Hence, it makes ICPG more scalable and flexible for pruning in diverse real-world graph learning tasks or applications. In addition, we also conduct extensive experiments to verify this point.

4.2 Inductive Co-Pruning Strategy

Here we present Inductive Co-Pruning of GNNs (ICPG) to learn the GLTs. Fig.1 demonstrates its overview, which consists of the following two steps:

Step 1: Co-Training AutoMasker and the GNN Model of Interest. Given an input graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$, AutoMasker first generates the edge mask $\boldsymbol{s}_{\mathcal{G}}$ via (4). Then we apply $\boldsymbol{s}_{\mathcal{G}}$ to the adjacency matrix $\boldsymbol{A}$ to create the soft-masked graph $\mathcal{G}_s = (\boldsymbol{s}_{\mathcal{G}} \odot \boldsymbol{A}, \boldsymbol{X})$, which fully reflects AutoMasker's decision for the importance of each edge, such that less important edges are prone to have lower mask values. Finally, we feed the soft-masked graph into the GNN model to co-train the AutoMasker and the model. The GNN model adopts the masked graph to learn the representation and make predictions, which can be viewed as the supervision signals to guide the AutoMasker to achieve a more accurate decision. The detailed co-training process is shown in Algorithm 1. When the training is done, we conduct Step 2 to perform the pruning.

Algorithm 2 Finding GLTs by ICPG

```
Input: Graphs $\mathcal{D} = \{\mathcal{G}_i = (\boldsymbol{A}_i, \boldsymbol{X}_i)\}_{i=1}^N$, $f(\cdot, \boldsymbol{\Theta}_{g_0})$,
    $AutoMasker(\cdot, \boldsymbol{\Theta}_{a_0})$, sparsity levels $s_d$, $s_\theta$.
Output: GLT $\{\mathcal{G}'_i = (\boldsymbol{m}_{\mathcal{G}_i} \odot \boldsymbol{A}_i, \boldsymbol{X}_i)\}_{i=1}^N$, $f(\cdot; \boldsymbol{m}_\Theta \odot$
    $\boldsymbol{\Theta}_{g_0})$.
Initialize masks set $\mathcal{M} \leftarrow \{\boldsymbol{m}_{\mathcal{G}_i} \leftarrow \boldsymbol{A}_i\}_{i=1}^N$
Initialize GNN mask $\boldsymbol{m}_\Theta \leftarrow \mathbf{1} \in \mathbb{R}^{\|\boldsymbol{\Theta}_{g_0}\|_0}$
while the sparsity of $\mathcal{M} < s_d$, $\boldsymbol{m}_\Theta < s_\theta$ do
    Sparsify GNN $f(\cdot; \boldsymbol{\Theta}_{g_0})$ with $\boldsymbol{m}_\Theta$ and graphs
    $\{\mathcal{G}_i = (\boldsymbol{A}_i, \boldsymbol{X}_i)\}_{i=1}^N$ with the mask set $\mathcal{M}$ and
    get the new masks as presented in Algorithm 1.
    Update $\mathcal{M} \leftarrow \{\boldsymbol{m}_{\mathcal{G}_i} \leftarrow \boldsymbol{m}'_{\mathcal{G}_i}\}_{i=1}^N$
    Update $\boldsymbol{m}_\Theta \leftarrow \boldsymbol{m}'_\Theta$
    Rewind AutoMasker's weight to $\boldsymbol{\Theta}_{a_0}$.
    Rewind GNN's weight to $\boldsymbol{\Theta}_{g_0}$.
end while
```

Step 2: Co-Sparsifying the Input Graphs and GNN Model. Having obtained the well-trained AutoMasker and GNN, we can apply the learned knowledge to co-sparsify the graphs and model. For graphs, AutoMasker predicts the importance score (e.g., mask value) for each edge. Then the edges of a certain graph are sorted based on their mask values, and the edges with p_g ratio of the lowest-masks are pruned to obtain the mask $\boldsymbol{m}_{\mathcal{G}}$. For GNN, we sort the parameters based on the their magnitude and prune p_θ ratio of the lowest-magnitude parameters to obtain the binary model mask $\boldsymbol{m}_\Theta$. Under the current sparsity, we now successfully obtain the sparsified graph $\mathcal{G}' = (\boldsymbol{m}_\mathcal{G} \odot \boldsymbol{A}, \boldsymbol{X})$ and the sparsified GNN mask $\boldsymbol{m}_\Theta$. Finally, we need to check whether the sparsity meets our condition. If the sparsity is satisfied, the algorithm is completed; if not, we reuse the found GLT to update the original graphs and GNN model, and iteratively run Step 1 and Step 2 (dotted arrow in Fig.1) until the condition is met.

In summary, Algorithm 2 offers the detailed process of ICPG, where the sparsity levels s_θ and s_d refer to the proportions of model weights and graph edges that need to be pruned. Following LTH [8] and UGS [7], we also adopt an iterative pruning strategy to locate GLTs. In Algorithm 2, it will conduct Algorithm 1 to prune a certain proportion of graph edges and model weights. In our experiments, for graph data, we prune 5% of the edges each time, and for the model, we prune 20% of the weights each time. So we need to execute Algorithm 1 several times to achieve the given sparsity levels.

5 Experiments

In this section, we conduct extensive experiments to validate the effectiveness of ICPG. We first introduce the experimental settings and explore the existence of GLTs in graph classification and node classification. Then, we demonstrate the practicability, such as transferability, performance, and computational cost saving. Finally, more ablation studies and visualizations are provided.

5.1 Experimental Settings

Datasets. For graph classification, we adopt TU datasets [28–30], including biological graphs (NCI1, MUTAG), social graphs (COLLAB, RED-B, RED-M5K, RED-M12K). We also use superpixel graphs (MNIST, CIFAR-10) [31, 32], and Open Graph Benchmark (ogbg-ppa and ogbg-code2) [6]. We use these graph classification datasets for inductive graph learning. For node classification, we choose a transductive learning dataset, Cora, and an inductive learning dataset, PPI. The detailed statistics of the datasets are shown in Table 2, where "#" refers to the number and "Avg." means the average number.

Models. We adopt the same model architecture for the GNN backbone and GNN encoder in AutoMasker. For graph classification tasks and Cora, we adopt the GCN [1] model. For PPI, we choose GAT [2] to achieve a better baseline performance.

Table 2. Datasets Statistics

Dataset	#Graph	Avg. Nodes	Avg. Edges	Avg. Degree	#Class
NCI1	4,110	29.87	32.30	1.08	2
MUTAG	188	17.93	19.79	1.10	2
COLLAB	5,000	74.49	2457.78	32.99	3
RED-B	2,000	429.63	494.07	1.15	2
RED-M5K	4,999	508.52	594.87	1.17	5
RED-M12K	11,929	391.41	456.89	1.16	11
MNIST	70,000	70.57	564.56	8.00	10
CIFAR-10	60,000	117.63	941.04	8.00	10
ogbg-ppa	158,100	243.40	2,266.10	9.31	37
ogbg-code2	452,741	125.20	124.20	0.99	-
Cora	1	2708.00	5429.00	2.00	7
PPI	24	2372.67	34113.16	14.38	121

Training Settings. Here we provide the detailed training settings of the proposed ICPG. All training hyper-parameters such as epoch, learning rate (LR), optimizer, batch size, and weight decay are summarized in Table 3. For the devices, we adopt the NVIDIA GeForce RTX 3090 (24GB GPU) to conduct all our experiments. To help readers easily reproduce our results, we also provide the code of our work[1].

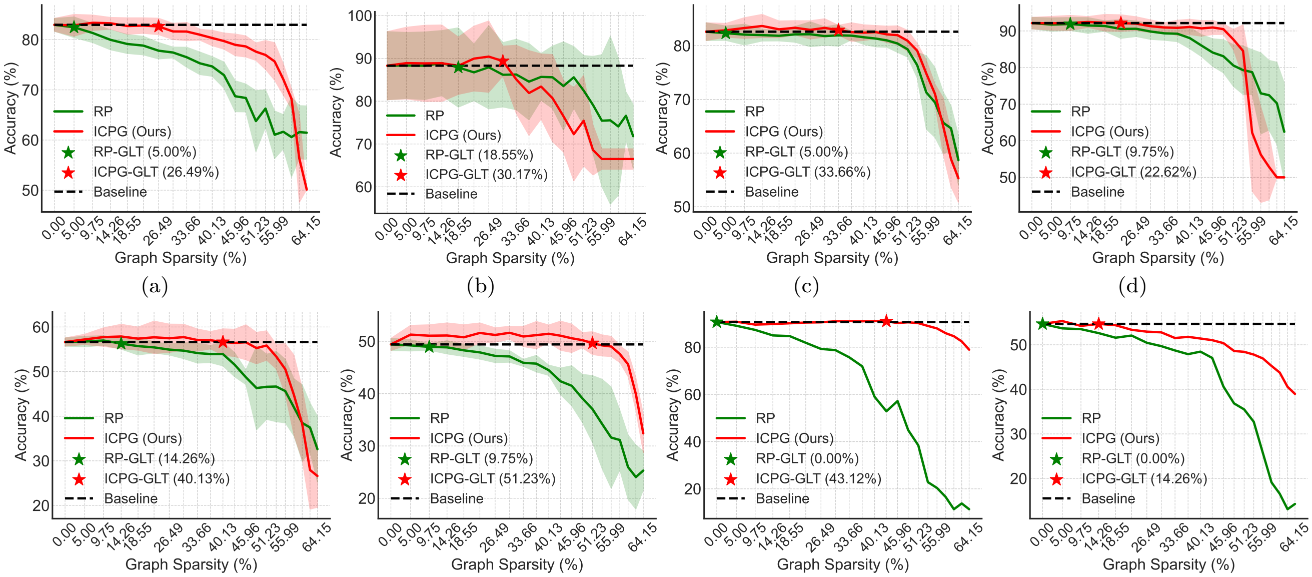
Table 3. Training Details of ICPG

Dataset	Epoch	LR	Optimizer	Batch Size	Weight Decay
TU	100	0.001	Adam	128	0
Superpixel	100	0.001	Adam	128	0
ppa	100	0.001	Adam	32	0
code2	25	0.001	Adam	128	0
Cora	200	0.01	Adam	1	0.0005
PPI	100	0.005	Adam	1	0

5.2 GLTs in Graph Classification Tasks

We first conduct experiments to find the GLTs in graph classification tasks. The results of different graph sparsity-levels are displayed in Fig.2 and Fig.3. Due to the limited space, we omit the results of weight sparsity,

[1] https://github.com/yongduosui/ICPG, Apr. 2023



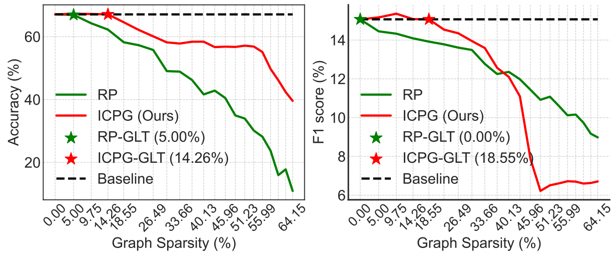
(a) (b) (c) (d)

(e) (f) (g) (h)

Fig.2. Graph classification performance across different graph sparsity-levels. (a) NCI1. (b) MUTAG. (c) COLLAB. (d) RED-B. (e) RED-M5K. (f) RED-M12K. (g) MNIST. (h) CIFAR-10.

which follow a similar trend. We also plot the random pruning (RP) for comparison. Stars denote extreme sparsity, which is the maximal sparsity-level without performance degradation. We make the following observations.

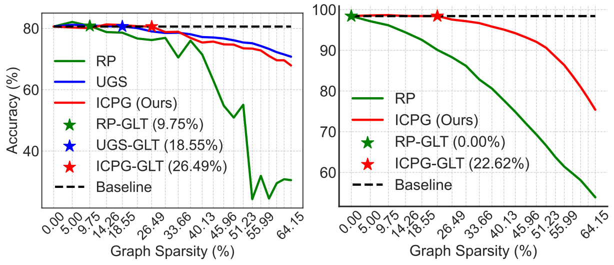
Observation 1. GLTs extensively exist in graph classification tasks. Utilizing the ICPG, we successfully locate the GLTs with different sparsity-levels from diverse types of graphs. For NCI1 and MUTAG, we precisely identify GLTs with extreme graph sparsity at 26.49% and 30.17%, GNN sparsity at 73.79% and 79.03%, respectively. On four social network datasets, we find the GLTs with graph sparsity of 22.62%-51.23% and GNN sparsity-level of 67.23%-95.60%. For MNIST and CIFAR-10, the GLTs are achieved with graphs sparsity of 43.13% and 14.26%, and GNN sparsity of 91.41% and 48.80%. These results show that ICPG can inductively locate the high-quality GLTs with different graph types, and demonstrate the potential of efficient training and inference with sparser graphs and lightweight GNNs without sacrificing performance.



(a) (b)

Fig. 3. Graph classification performance across different graph sparsity-levels on large-scale datasets. (a) ogbg-ppa. (b) ogbg-code2.

Observation 2. AutoMasker has good generalization ability. The mainstream graph sparsification techniques [7,17,19] cannot inductively prune unseen graphs. However, the AutoMasker can flexibly overcome this challenge. Compared with RP, ICPG can find more sparse subgraphs and subnetworks and keep a large gap with RP. For instance, the RED-M5K and RED-M12K graphs pruned by ICPG can achieve 40.13% and 51.23% extreme graph sparsity, improving 25.87% and 41.48% compared with RP, which keeps an extremely large superiority. These indicate that AutoMasker can precisely capture more significant core-patterns from the training graphs and have a good generalization ability to predict the high-quality masks for unseen graphs.



(a) (b)

Fig. 4. Transductive and inductive node classification performance across different graph sparsity-levels. (a) Cora. (b) PPI.

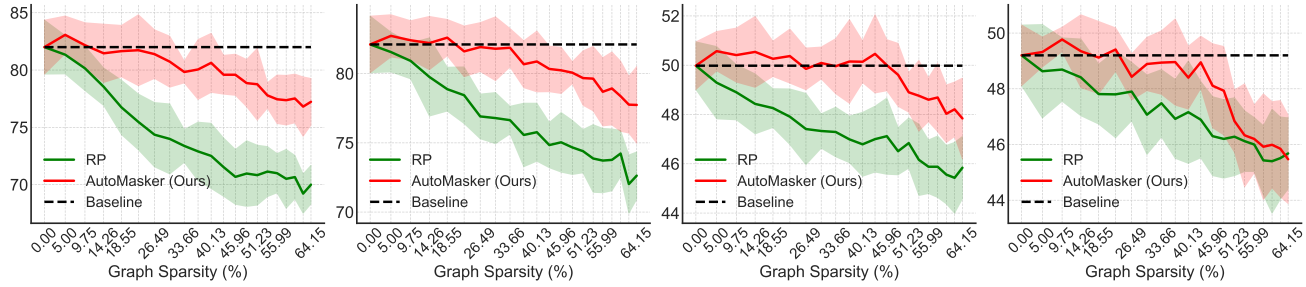
Observation 3. The extreme sparsity of GLTs depends on the property of the graphs. Although ICPG achieves higher sparsity than RP on most graphs, the improvements are not obvious on a small part of the graphs, such as biochemical molecule graphs NCI1 and MUTAG. We give the following explanations. Firstly, most of the edges in these graphs are important, such as a certain edge may correspond to a crucial chemical bond, which may drastically affect the chemical properties of the molecule if pruned. Furthermore, the graph size is relatively small, which just includes a few dozen nodes and edges, therefore it is more sensitive to pruning. The study GraphCL [33] also finds a similar phenomenon with us. It states that the performance of these chemical and molecular datasets could not be improved by data augmentation. In our work, we also experimentally demonstrate the phenomenon that edges in datasets of biochemical molecules, are more important than those of social networks. For example, on the biochemical molecule datasets, ICPG can achieve an average extreme sparsity of 28.6%, while on the social network datasets, ICPG can achieve an average extreme sparsity of 36.9%.

Observation 4. AutoMasker can well tackle larger-size and larger-quantity graphs. Fig.3 demonstrates the results on the challenging OGB datasets, which consist of larger-size graphs (2266.1 edges and 243.4 nodes on average per graph for ogbg-ppa) and larger-quantity graphs (452,741 graphs for ogbg-code2). We surprisingly find that the OGB datasets are so intractable that RP can only locate 5% graph sparsity-level of GLT on the ogbg-ppa, and it is even impossible to find any sparser GLTs on the ogbg-code2. Despite this, the proposed ICPG can locate the GLTs with 14.26% and 18.55% graph sparsity-levels, 48.80% and 59.40% GNN sparsity-level on ogbg-ppa and ogbg-code2, respectively. The superior performance further verifies the generalization ability and strong scalability.

5.3 GLTs in Node Classification Tasks

Since ICPG can achieve excellent performance on diverse types and scales of graphs, we also want to explore if it can also tackle node-level tasks. To answer this question, we conduct experiments on Cora and PPI, which are commonly used in transductive and inductive node classification tasks. We also reproduce the recent work ADMM [19,22] and UGS [7] for Cora (cannot apply for inductive setting) for comparison. From the results in Fig.4, we give the following observations.

Observation 5. ICPG achieves excellent performance in node classification tasks. Firstly, for Cora, all pruning methods consistently outperform RP and keep a large gap as the sparsity-level increases. UGS just adopts simple trainable masks for edges without considering the global topological structure of the entire graph. ADMM only optimizes the adjacency matrix without considering the GNN model. ICPG overcomes these two issues, thereby predicting more high-quality masks. Hence, ICPG can locate sparser GLTs than ADMM (↑21.49%) and UGS (↑7.94%). Secondly, the performance of ICPG drops faster in the later stage. These phenomena also exist in several other datasets, such as ogbg-code2, RED-B and RED-M5K. From Algorithm 2, each round of ICPG will preferentially prune the model weights and graph edges with the lowest importance score, therefore those unimportant weights and edges will be removed at an early stage. Some recent studies [34–36] have also demonstrated that there exist important features in graph data, often called causal subgraphs [35] or rationales [34,36]. These



(a) (b) (c) (d)

Fig.5. Performance of diverse GNNs on the sparse graphs pruned by AutoMasker. (a) NCI1 (GIN). (b) NCI1 (GAT). (c) RED-M12K (GIN). (d) RED-M12K (GAT).

Table 4. Graph Classification Accuracy (%)

Dataset	Method	Graph Sparsity					
		0% (No pruning)	9.75%	18.55%	33.66%	45.96%	55.99%
RED-B	RP	92.15	90.60	89.75	86.75	85.15	85.34
	AutoMasker	92.15	**92.16**	**91.05**	**90.15**	**90.06**	**89.64**
RED-M5K	RP	56.63	56.33	55.85	54.81	54.19	54.95
	AutoMasker	56.63	**56.89**	**56.69**	**57.01**	**56.97**	**56.09**

features often determine the intrinsic property of the graph data, such as the functional groups in molecular data, or some important edge collections in social networks [35]. Perturbing or pruning them may greatly affect performance. Based on our results, ICPG tends to remove the redundant parts of the data in the early stage, and the remaining parts are basically the causal features of the graph data. Intuitively, the visualization results in Subsection 5.7 can also prove this point. Hence, the performance of ICPG will drop significantly in later stages.

5.4 Transferability of the AutoMasker

We consider two perspectives to verify the transferability of the AutoMasker: GNN-level transferability and graph-level transferability. From GNNs view, we transfer the sparse graphs pruned by AutoMasker to the other two popular GNN models: GIN [9] and GAT [2]. From graphs view, we first pre-train the AutoMasker on the larger-scale social dataset RED-M12K and then transfer the well-trained AutoMasker to prune the other two smaller-scale social datasets: RED-B and RED-M5K. We keep the GNN models unpruned on transferred tasks. The performance of graph classification over different sparsity-levels are provided in Fig. 5 and Table 4. The best results are shown in bold. We make the following observations.

Observation 6. AutoMasker has both GNN-level and graph-level transferability. For GNN-level, we observe from Fig. 5 that GIN and GAT achieve ranging 9.75%-45.96% and 18.55%-22.62% sparsity on NCI1 and RED-M12K without sacrificing performance. AutoMasker also outperforms RP and keeps a large gap. These results demonstrate that AutoMasker can effectively extract the model-agnostic subgraphs. These subgraphs contain significant semantic information and can be universally transferred to diverse GNN architectures without performance degradation. As for graph-level transferability in Table 4, the classification accuracy of random pruning decreases as the sparsity-level increases. For RED-B and RED-M5K, when the sparsity-level increases from 0 to 55.99%, the ac-

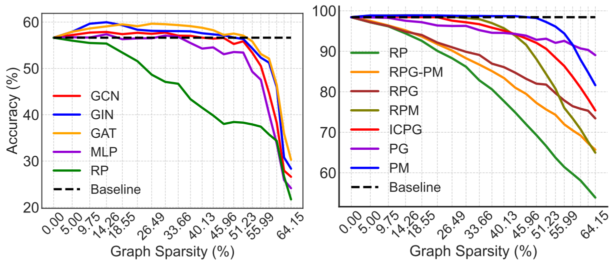
Table 5. Graph Classification Accuracy (%) on NCI1, COLLAB and RED-M5K Datasets across Different Sparsity-Levels

Method	NCI1			COLLAB			RED-M5K		
	22.62%	33.66%	40.13%	22.62%	33.66%	40.13%	22.62%	33.66%	40.13%
GraphSAGE [16]	76.62	72.97	71.27	73.93	69.04	68.50	47.65	44.25	36.83
DropEdge [18]	82.24	81.40	80.14	82.16	81.96	81.52	50.37	46.85	45.35
NeuralSparse [33]	81.43	80.34	79.83	81.63	77.76	75.35	52.82	51.56	49.95
ICPG (ours)	**82.82**	**81.63**	**80.34**	**83.34**	**82.90**	**82.44**	**57.69**	**57.07**	**56.63**
Improvement (ours)	↑0.58%	↑0.23%	↑0.20%	↑1.18%	↑0.94%	↑0.92%	↑4.87%	↑5.51%	↑6.68%

Table 6. Inference MACs Comparisons

Method	MUTAG	NCI1	COLLAB	RED-B	RED-5K	RED-12K	ogbg-code2	ogbg-ppa
Baseline	23.53M	834.97M	3445.43M	4723.60M	13661.66M	24366.16M	1397.95G	5680.79G
ICPG (ours)	5.09M	223.76M	1103.06M	1583.97M	1584.23M	1533.57M	672.91G	2869.69G
Reduction (ours)	↓ 78.36%	↓ 73.20%	↓ 67.98%	↓ 66.47%	↓ 88.40%	↓ 93.71%	↓ 51.86%	↓ 49.48%

curacy decreases by 7.39% and 2.97%, respectively. While AutoMasker can achieve consistent improvement within all sparsity-levels. Furthermore, the GNN model trained with more sparse graphs even outperforms the GNN trained with the original dense graphs, such as RED-B at 9.75% and RED-M5K at 9.75%-45.96%. It demonstrates that AutoMasker can well transfer the knowledge from large-scale upstream tasks to small-scale downstream tasks and achieve a double-win: with lower computational cost and better performance. In summary, AutoMasker can learn model-agnostic, general, and significant sparse subgraph structures from the graphs, so that it has outstanding GNN-level and graph-level transferability.



(a) (b)

Fig.6. (a) The comparison of different encoders in AutoMasker on RED-M5K dataset. (b) The comparison of each component in ICPG on PPI dataset.

5.5 Performance and Inference MACs.

Performance Comparison. To demonstrate the practicability of ICPG, we validate the performance of the GLTs. We adopt GraphSAGE [21], DropEdge [18], and NeuralSparse [17], which can achieve graph sparsification inductively. For a fair comparison, we adjust the hyper-parameters in GraphSAGE (sampling rate) and DropEdge (dropping rate) to achieve similar sparsity-levels. In Table 5, we observe that our methods consistently outperform other baselines at all sparsity-levels. It demonstrates the superiority of ICPG.

Inference MACs. Following UGS [1], we translate the sparsity-level to the inference MACs reduction for evaluating the computational cost. We report the extreme inference MACs, which is the minimal MACs without performance degradation. The results are shown in Table 6. Compared with the full baseline, our method can significantly reduce the computational cost by about 51.86%-93.71% from small-scale (e.g., MUTAG) to large-scale (e.g., ogbg-code2) datasets, without sacrificing performance. These results further verify the practicability of ICPG.

5.6 Ablation Study

Encoder Networks. AutoMasker is designed on a GNN-based encoder, which leads to a global understanding of each edge from the entire graphs. Therefore, we extensively investigate the impact of diverse encoders, such as GNN-based or MLP-based encoders. We can observe the results from Fig.6 (a) that, for all the GNN-based encoders, AutoMasker can achieve good performance: 45.96% extreme sparsity for GCN and 51.23% for GIN and GAT, while MLP-based encoder only achieves 33.66% extreme sparsity. It indicates that the message-passing scheme of the GNN encoder naturally considers the graph structure from a global view, while the MLP-based encoder does not.

Co-sparsification. To study the effectiveness of each component in ICPG, we apply them to the graphs and model independently. We explore mask-based Pruning for Graphs (PG), magnitude-based Pruning for Model (PM), Random Pruning only for Graphs (RPG), only for Models (RPM), both of all (RP), Random Pruning for Graphs with magnitude-based Pruning for Model (RPG-PM). The results are summarized in Fig.6 (b). We can find that: PG can also find the matching subgraphs. PM can also locate the matching subnetworks at 14.26% sparsity, which is consistent with the LTH [8] in the computer vision field. ICPG significantly outperforms RP and RPG-PM, and the gap gradually widens as the sparsity increases. We also observe that ICPG is even better than PG (↑12.87%), and we make the following explanations.

(1) As for PG, with the sparsity-level gradually increasing, the graphs also become more simple. If we still train the over-parameterized GNN model with simple graphs, it may cause over-fitting.

(2) Slightly pruning the over-parameterized GNN through PM can be regarded as a kind of regularization, which will improve the performance, and it is consistent with LTH [8]. Further, the regularized GNN can additionally provide AutoMasker with more precise supervision signals from the gradient in backpropagation to make wise decisions. In summary, these results suggest the significance of co-training the AutoMasker and GNN, and co-sparsifying the input graphs and model to achieve better performance.

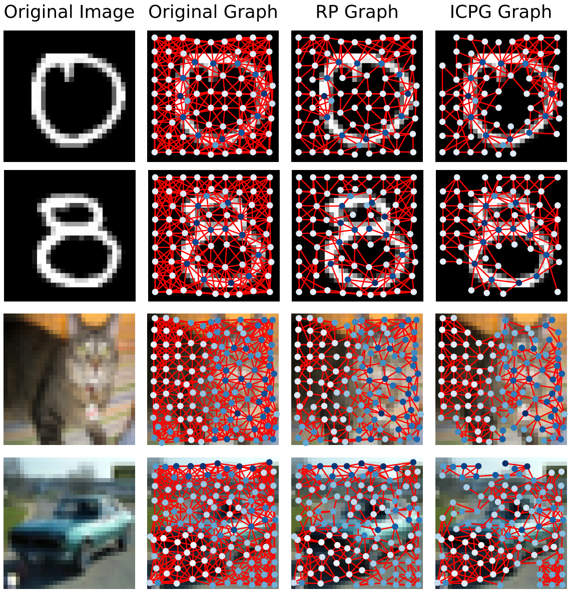


Fig.7. Visualization of the subgraphs extracted by AutoMasker from MNIST and CIFAR-10 superpixel graphs. Original images and original graphs are displayed on the first and second columns. The sparsity-level of RP and ICPG is 64.15%.

5.7 Visualization

To visualize the sparsifed subgraphs in GLTs, we select graphs with 64.15% sparsity-level from the MNIST and CIFAR-10 superpixel datasets. For better comparison, we also plot the original images, original graphs, random pruning graphs, which are depicted in Fig.7. We have the following findings.

For MNIST and CIFAR-10, the edges between nodes that locate on the digitals and object pixels (the dark blue nodes) should be denser, which are conducive to the graph classification tasks. RP evenly prunes the significant edges or structures without considering any important reference, which makes the core subgraphs destroyed and seriously deteriorates the performance. ICPG utilizes AutoMasker to learn the significance of each edge from a global view and can precisely prune redundant edges. As the MNIST ICPG graph, the pruned edges are mainly located on non-digital pixels, such as the upper-left, lower-right corners and the center part of the number 0 and the lower-left corner of the number 8, while the remaining edges or nodes are mainly located on digital pixels. These patterns further demonstrate the rationality and explainability of ICPG.

6 Conclusions

In this work, we endowed the graph lottery tickets with inductive pruning capacity. We proposed a simple but effective pruning framework ICPG, to co-sparsify the input graphs and GNN model. Our core innovation, the AutoMasker, leverages a global comprehension of edge significance based on the entire graph's topological structure. This ensures the creation of superior graph masks, exhibiting a strong generalization capability in inductive learning contexts. Through extensive experiments across various graph types, scales, learning settings, and tasks, we consistently established that ICPG can effectively accomplish high sparsity within both graph data and GNN models. This compelling evidence underscores ICPG's potential to effectively optimize the efficiency of GNN models.

In future work, we intend to refine ICPG's multi-round iterative pruning paradigm, investigating methods to enhance pruning efficiency. This advancement could significantly reduce computational costs during training, paving the way for more resource-efficient GNN models.

References

[1] Kipf, T. N., Welling, M. Semi-supervised classification with graph convolutional networks. In *Proc. the 5th International Conference on Learning Representations*, Apirl 2017.

[2] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y. Graph attention networks. In *Proc. the 6th International Conference on Learning Representations*, April 2018.

[3] Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018. https://arxiv.org/abs/1812.08434, Oct 2021.

[4] Li, G., Xiong, C., Thabet, A., Ghanem, B. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*. https://arxiv.org/abs/2006.07739, Jun 2020.

[5] Li, G., Müller, M., Qian, G., Perez, I. C. D., Abualshour, A., Thabet, A. K., Ghanem, B. Deepgcns: Making gcns go as deep as cnns. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* DOI: 10.1109/TPAMI.2021.3074057.

[6] Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. In*Proc. Advances in Neural Information Processing Systems*, December 2020, pp. 22118-22133.

[7] Chen, T., Sui, Y., Chen, X., Zhang, A., Wang, Z. A unified lottery ticket hypothesis for graph neural networks. In *Proc. the 38th International Conference on Machine Learning*, July 2021, pp. 1695-1706.

[8] Frankle, J., Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *Proc. the 7th International Conference on Learning Representations*, May 2019.

[9] Xu, K., Hu, W., Leskovec, J., Jegelka, S. How powerful are graph neural networks? In *Proc. the 7th International Conference on Learning Representations*, May 2019.

[10] Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, December 2018, pp.4800–4810.

[11] Chen, T., Frankle, J., Chang, S., Liu, S., Zhang, Y., Carbin, M., Wang, Z. The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2021, pp. 16306-16316.

[12] Ma, H., Chen, T., Hu, T.-K., You, C., Xie, X., Wang, Z. Spending Your Winning Lottery Better After Drawing It. *arXiv*, abs/2101.03255. https://arxiv.org/abs/2101.03255, Oct 2021.

[13] Liu, Z., Sun, M., Zhou, T., Huang, G., Darrell, T. Rethinking the value of network pruning. In *Proc. the 7th International Conference on Learning Representations*, Apirl 2019.

[14] Wang, C., Zhang, G., Grosse, R. Picking winning tickets before training by preserving gradient flow. In *8th International Conference on Learning Representations*, Apirl 2020.

[15] Savarese, P., Silva, H., Maire, M. Winning the lottery with continuous sparsification. In *Proc. Advances in Neural Information Processing Systems 33 pre-proceedings*, December 2020, pp. 11380-11390.

[16] Voudigari, E., Salamanos, N., Papageorgiou, T., Yannakoudakis, E. J. Rank degree: An efficient algorithm for graph sampling. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, August 2016, pp. 120–129. DOI: 10.1109/ASONAM.2016.7752223.

[17] Leskovec, J., Faloutsos, C. Sampling from large graphs. In *Proc. the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, August 2006, pp. 631–636, DOI: 10.1145/1150402.1150479.

[18] Zeng, H., Zhou, H., Srivastava, A., Kannan, R., Prasanna, V. Graphsaint: Graph sampling based inductive learning method. In *Proc. the 8th International Conference on Learning Representations*, Apirl 2020.

[19] Franceschi, L., Niepert, M., Pontil, M., He, X. Learning discrete structures for graph neural networks. In *International Conference on Machine Learning*, June 2019, pp. 1972–1982.

[20] Ying, R., Bourgeois, D., You, J., Zitnik, M., Leskovec, J. GNNExplainer: Generating Explanations for Graph Neural Networks. *arXiv preprint arXiv:1903.03894*, https://arxiv.org/abs/1903.03894, Nov 2019.

[21] Ye, Y., Ji, S. Sparse graph attention networks. *IEEE Transactions on Knowledge and Data Engineering*, pp.1–1, 2021. DOI:10.1109/TKDE.2021.3072345.

[22] Zheng, C., Zong, B., Cheng, W., Song, D., Ni, J., Yu, W., Chen, H., Wang, W. Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*, April 2020, pp.11458–11468.

[23] Rong, Y., Huang, W., Xu, T., Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *Proc. the 8th International Conference on Learning Representations*, April 2020.

[24] Li, J., Zhang, T., Tian, H., Jin, S., Fardad, M., Zafarani, R. Sgcn: A graph sparsifier based on graph convolutional networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2020, pp. 275–287.

[25] Chen, J., Ma, T., Xiao, C. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *Proc. the 6th International Conference on Learning Representations*, April 2018.

[26] Hamilton, W., Ying, Z., Leskovec, J. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, December 2017, pp.1024–1034.

[27] You, H., Lu, Z., Zhou, Z., Fu, Y., Lin, Y. Early-bird gcns: Graph-network co-optimization towards more efficient gcn training and inference via drawing early-bird lottery tickets. *arXiv preprint arXiv:2103.00794*, https://arxiv.org/abs/2103.00794, Dec 2021.

[28] Chen, T., Bian, S., Sun, Y. Are powerful graph neural nets necessary? a dissection on graph classification. *arXiv preprint arXiv:1905.04579*, https://arxiv.org/abs/1905.04579, Jun 2020.

[29] Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., Bresson, X. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, https://arxiv.org/abs/2003.00982, Dec 2022.

[30] Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., Neumann, M. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, https://arxiv.org/abs/2007.08663, Jul 2020.

[31] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 11, pp. 2274-2282, Nov. 2012,* DOI: 10.1109/TPAMI.2012.120.

[32] Knyazev, B., Taylor, G. W., Amer, M. R. Understanding attention and generalization in graph neural networks. In *Proc. Advances in Neural Information Processing Systems*, 2019.

[33] You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., Shen, Y. Graph contrastive learning with augmentations. In *Proc. Advances in Neural Information Processing Systems*, December 2020, pp. 5812-5823.

[34] Wu, Y., Wang, X., Zhang, A., He, X., Chua, T. Discovering invariant rationales for graph neural networks. In *Proc. the 10th International Conference on Learning Representations*, April 2022.

[35] Sui, Y., Wang, X., Wu, J., He, X., Chua, T. Causal attention for interpretable and generalizable graph classification. In *Proc. the 28th ACM SIGKDD international conference on Knowledge discovery and data mining*, August 2022, pp. 1696–1705. DOI: 10.1145/3534678.3539366.

[36] Liu, G., Zhao, T., Xu, J., Luo, T., Jiang, M. Graph rationalization with environment-based augmentations. In *Proc. the 28th ACM SIGKDD international conference on Knowledge discovery and data mining*, August 2022, pp. 1069–1078. DOI: 10.1145/3534678.3539347.

Yong-Duo Sui is currently pursuing his Ph.D. degree in computer science from University of Science and Technology of China (USTC), Hefei. His research interests include graph learning, out-of-distribution generalization, sparse neural networks.

Xiang Wang received his Ph.D. degree from the National University of Singapore, in 2019. He is currently a professor with the University of Science and Technology of China, Hefei. His research interests include recommender systems, graph learning, and explainable deep learning techniques.

Tianlong Chen is a Ph.D. candidate in Electrical and Computer Engineering at University of Texas at Austin. He received his B.S. degree from University of Science and Technology of China, Hefei in 2017. He has also interned at IBM Research, Facebook AI, Microsoft Research and Walmart Technology Lab. His research focuses sparse neural networks, AutoML, adversarial robustness, self-supervised learning, and graph learning.

Meng Wang received the B.E. and Ph.D. degrees from the Special Class for the Gifted Young, Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei in 2003 and 2008, respectively. He worked as an Associate Researcher at Microsoft Research Asia and a Senior Research Fellow at the National University of Singapore. He is a Professor at the Hefei University of Technology. His current research interests include multimedia content analysis, computer vision, and pattern recognition.

Xiang-Nan He received his Ph.D. degree in computer science from National University of Singapore, in 2016. He is now a professor with the University of Science and Technology of China, Hefei. His research interests include information retrieval, data mining, and multi-media analytics.

Tat-Seng Chua received the Ph.D. degree from the University of Leeds, U.K. He is the KITHCT chair professor with the School of Computing, National University of Singapore. He was the acting and founding dean of the School during 1998-2000. His main research interests include multimedia information retrieval and social media analytics. In particular, his research focuses on the extraction, retrieval and question-answering (QA) of text and rich media arising from the Web and multiple social networks.