A1. More Implementation Details

Datasets Download Links. As for small-scale datasets, we take the commonly used semi-supervised node classification graphs: Cora, Citeseer and pubMed. For larger-scale datasets, we use three Open Graph Benchmark (OGB) (Hu et al., 2020) datasets: Ogbn-ArXiv, Ogbn-Proteins and Ogbl-Collab. All the download links of adopted graph datasets are included in Table A3.

Table A3. Download links of graph datasets.

Dataset	Download links and introduction websites
Cora	https://linqs-data.soe.ucsc.edu/public/lbc/cora.tgz
Citeseer	https://linqs-data.soe.ucsc.edu/public/lbc/citeseer.tgz
PubMed	https://linqs-data.soe.ucsc.edu/public/Pubmed-Diabetes.tgz
Ogbn-ArXiv	https://ogb.stanford.edu/docs/nodeprop/#ogbn-arxiv
Ogbn-Proteins	https://ogb.stanford.edu/docs/nodeprop/#ogbn-proteins
Ogbl-Collab	https://ogb.stanford.edu/docs/linkprop/#ogbl-collab

Train-val-test Splitting of Datasets. As for node classification of small- and medium-scale datasets, we use 140 (Cora), 120 (Citeseer) and 60 (PubMed) labeled data for training, 500 nodes for validation and 1000 nodes for testing. As for link prediction task of small- and medium-scale datasets Cora, Citeseer and PubMed, we random sample 10% edges as our testing set, 5% for validation, and the rest 85% edges are training set. The training/validation/test splits for Ogbn-ArXiv, Ogbn-Proteins and Ogbl-Collab are given by the benchmark (Hu et al., 2020). Specifically, as for Ogbn-ArXiv, we train on the papers published until 2017, validation on those published in 2018 and test on those published since 2019. As for Ogbn-Proteins, we split the proteins nodes into training/validation/test sets according to the species which the proteins come from. As for Ogbl-Collab, we use the collaborations until 2017 as training edges, those in 2018 as validation edges, and those in 2019 as test edges.

More Details about GNNs. As for small- and medium-scale datasets Cora, Citeseer and PubMed, we choose the two-layer GCN/GIN/GAT networks with 512 hidden units to conduct all our experiments. As for large-scale datasets Ogbn-ArXiv, Ogbn-Proteins and Ogbl-Collab, we use the ResGCN (Li et al., 2020a) with 28 GCN layers to conduct all our experiments. As for Ogbn-Proteins dataset, we also use the edge encoder module, which is a linear transform function in each GCN layer to encode the edge features. And we found if we also prune the weight of this module together with other weight, it will seriously hurt the performance, so we do not prune them in all of our experiments.

Training Details and Hyper-parameter Configuration. We conduct numerous experiments with different hyper-parameter, such as iterations, learning rate, γ_1, γ_2, and we choose the best hyper-parameter configuration to report the final results. All the training details and hyper-parameters are summarzed in Table A4. As for Ogbn-Proteins dataset, due to its too large scale, we use the commonly used random sample method (Li et al., 2020a) to train the whole graph. Specifically, we random sample ten subgraphs from the whole graph and we only feed one subgraph to the GCN at each iteration. For each subgraph, we train 10 iterations to ensure better convergence. And we train 100 epochs for the whole graph (100 iterations for each subgraph).

Evaluation Details We report the test accuracy/ROC-AUC/Hits@50 according to the best validation results during the training process to avoid overfitting. All training and evaluation are conducted for one run. As for the previous state-of-the-art method ADMM (Li et al., 2020b), we use the same training and evaluation setting as the original paper description, and we can reproduce the similar results compared with original paper.

Computing Infrastructures We use the NVIDIA Tesla V100 (32GB GPU) to conduct all our experiments.

A2. More Experiment Results

A2.1. Node Classification on Small- and Medium-scale Graphs with shallow GNNs

As shown in Figure A9, we also provide extensive results over GNN sparsity of GCN/GIN/GAT on three small datasets, Cora/Citeseer/PubMed. We observe that UGS finds the graph wining tickets at a range of GNNs sparsity from $20\% \sim 90\%$ without performance deterioration, which significantly reduces MACs and the storage memory during both training and inference processes.

A2.2. Link Prediction on Small- and Medium-scale Graphs with shallow GNNs

More results of link prediction with GCN/GIN/GAT on Cora/Citeseer/PubMed datasets are shown in Figure A10. We observe the similar phenomenon as the node classification task: using our proposed UGS can find the graph wining tickets at a range of graph sparsity from $5\% \sim 50\%$ and GNN sparsity from $20\% \sim 90\%$ without performance deterioration, which greatly reduce the computational cost and storage space during both training and inference processes.

A2.3. Large-scale Graphs with Deep ResGCNs

More results of larger-scale graphs with deep ResGCNs are shown in Figure A11. Results show that our proposed UGS can found GLTs, which can reach the non-trivial sparsity levels of graph $30\% \sim 50\%$ and weight $20\% \sim 80\%$ without performance deterioration.

Table A4. Implementation details of node classification and link prediction.

Task	Node Classification					Link Prediction			
Dataset	Cora	Citeseer	PubMed	Ogbn-ArXiv	Ogbn-Proteins	Cora	Citeseer	PubMed	Ogbn-Collab
Iteration	200	200	200	500	100	200	200	200	500
Learning Rate	8e-3	1e-2	1e-2	1e-2	1e-2	1e-3	1e-3	1e-3	1e-2
Optimizer	Admm	Admm	Admm	Admm	Admm	Admm	Admm	Admm	Admm
Weight Decay	8e-5	5e-4	5e-4	0	0	0	0	0	0
γ_1	1e-2	1e-2	1e-6	1e-6	1e-1	1e-4	1e-4	1e-4	1e-6
γ_2	1e-2	1e-2	1e-3	1e-6	1e-3	1e-4	1e-4	1e-4	1e-5

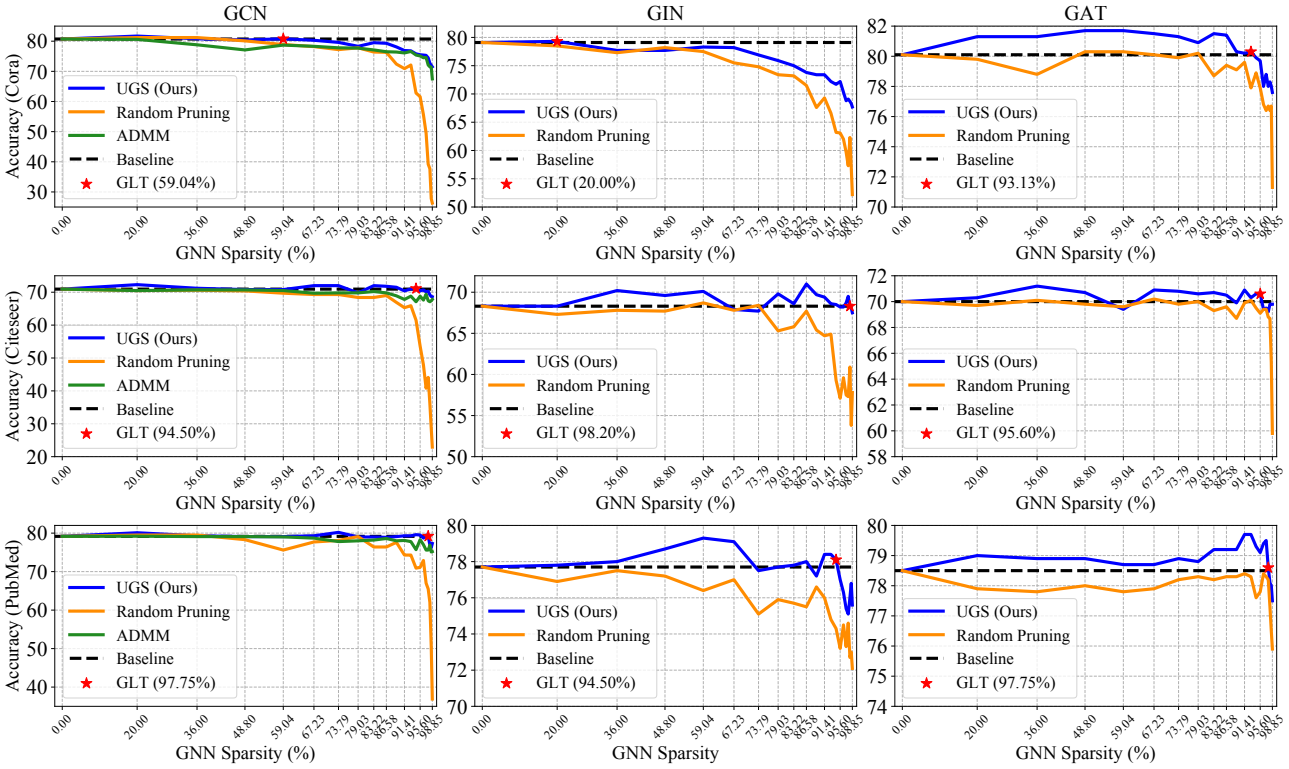

Figure A9. **Node classification** performance over achieved GNNs sparsity of GCN, GIN, and GAT on Cora, Citeseer, and PubMed datasets, respectively. *Red stars* (★) indicate the located GLTs, which reach comparable performance with extreme GNN sparsity. *Dash lines* represent the baseline performance of unpruned GNNs on full graphs.

A2.4. Graph Lottery Ticket with Pre-training

More results of node classification and link prediction on Cora and Citeseer dataset of GraphCL (You et al., 2020b) pre-training are shown in Figure A12. Results demonstrate that when using self-supervised pre-training, UGS can identify graph lottery tickets with higher qualities.

A2.5. More Analyses of Sparsified Graphs

As shown in Table A5, graph measurements are reported, including clustering coefficient, node and egde betweeness centrality. Results indicate that UGS seems to produce sparse graphs with more "critical" vertices which used to have more connections.

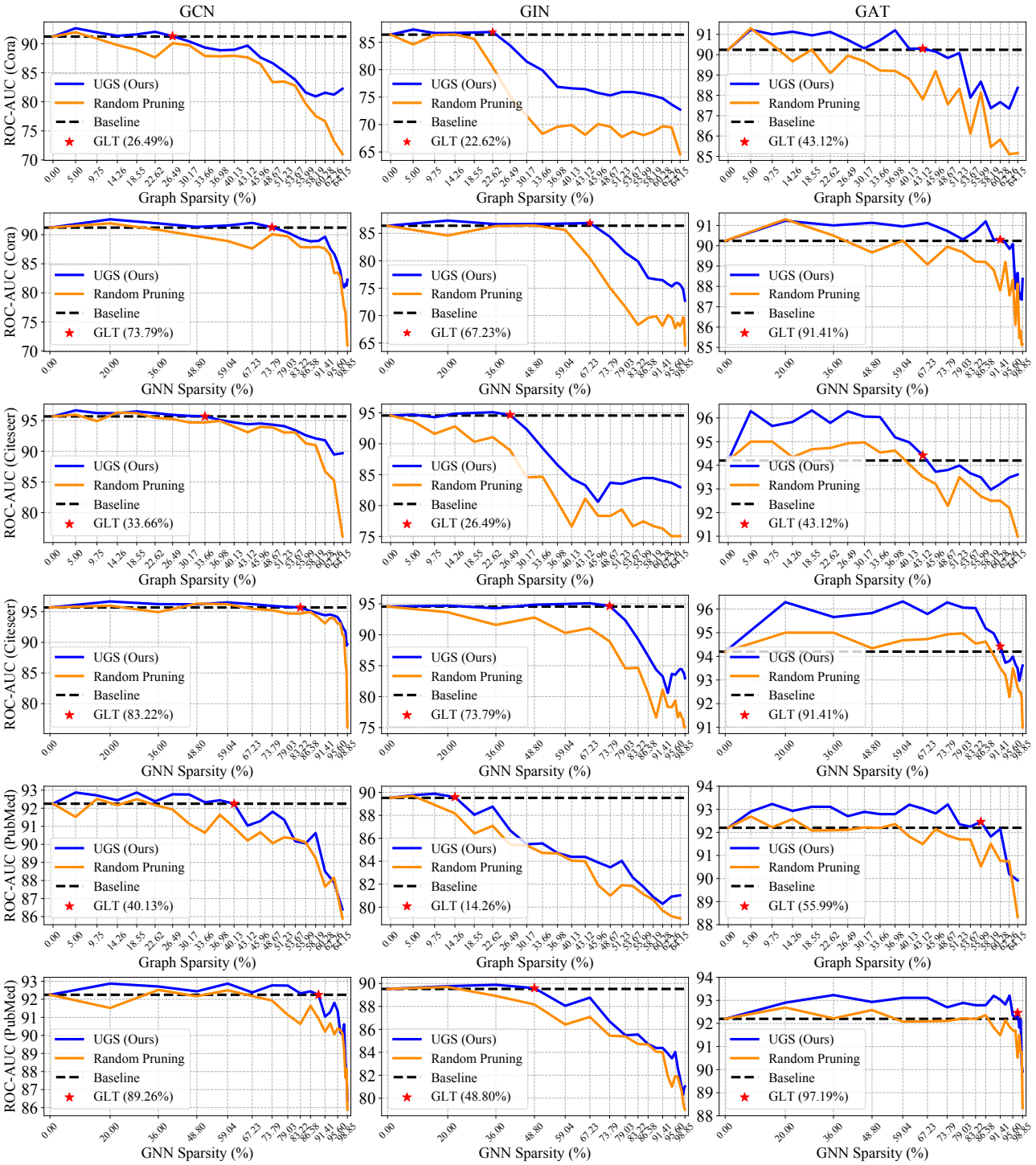

Figure A10. **Link prediction** performance over achieved GNNs sparsity of GCN, GIN, and GAT on Cora, Citeseer, and PubMed datasets, respectively. *Red stars* (★) indicate the located GLTs, which reach comparable performance with the extreme graph sparsity and GNN sparsity. *Dash lines* represent the baseline performance of unpruned GNNs on full graphs.

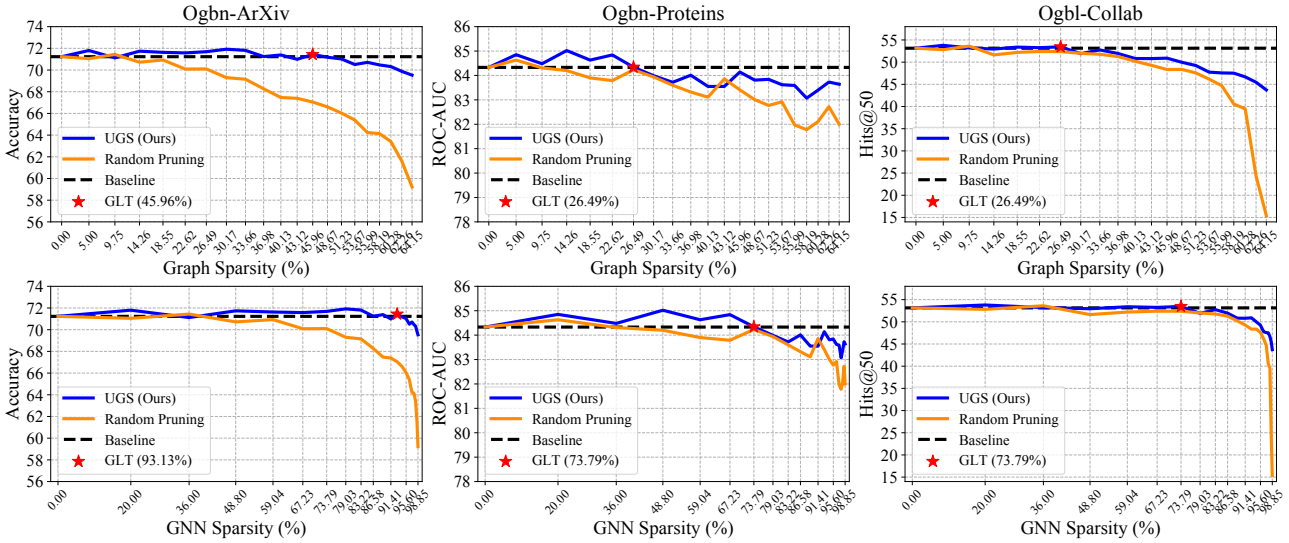

Figure A11. Node classification and link prediction performance over achieved graph sparsity and GNN sparsity of 28-layer **deep ResGCNs** on **large-scale graph** datasets.

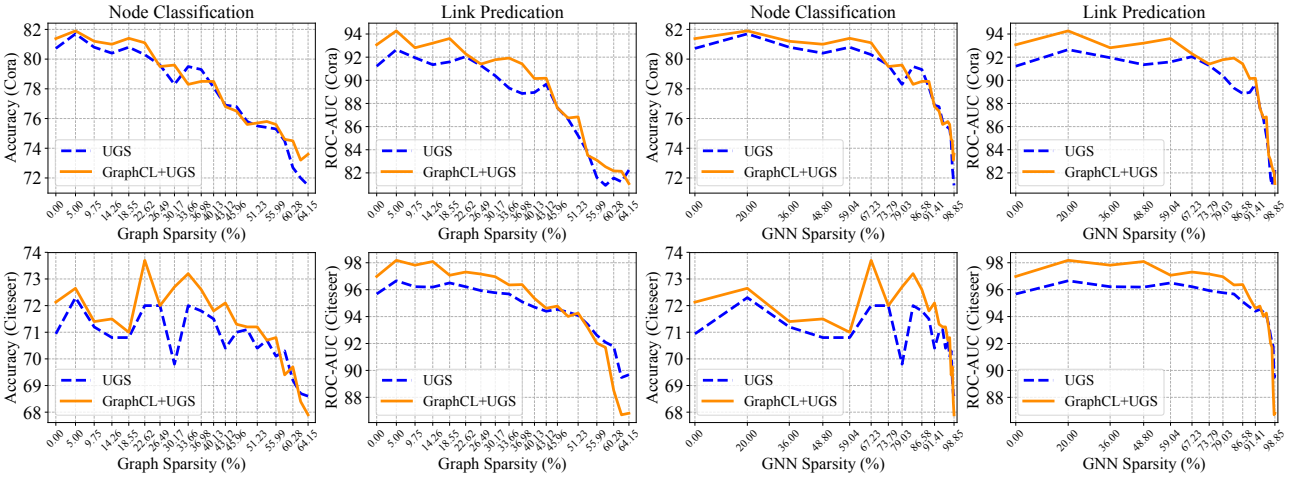

Figure A12. Drawing graph lottery tickets from randomly initialized and self-supervised pre-trained (GraphCL (You et al., 2020b)) GCNs on node classification (low label rate) and link prediction. Corresponding results over achieved graph and GNN sparsity are presented.

Table A5. Graph measurements of original graphs, sparse graphs from UGS, random pruning, and ADMM sparsification.

Measurements	Cora				Citeseer				PubMed			
	Original	UGS	RP	ADMM	Original	UGS	RP	ADMM	Original	UGS	RP	ADMM
Clustering Coefficient	0.14147	0.07611	**0.08929**	0.04550	0.24067	**0.15855**	0.15407	0.08609	0.06018	0.03658	**0.03749**	0.02470
Node Betweenness	0.00102	**0.00086**	0.00066	0.00021	0.00165	**0.00190**	0.00158	0.00122	0.00027	**0.00024**	0.00022	0.00018
Edge Betweenness	0.00081	**0.00087**	0.00068	0.00032	0.00101	**0.00144**	0.00123	0.00137	0.00014	**0.00019**	0.00015	0.00018