A Unified Lottery Ticket Hypothesis for Graph Neural Networks

[ICML 21] Tianlong Chen[1*], Yongduo Sui[2*], Xuxi Chen[1], Aston Zhang[3], Zhangyang Wang[1]
[1]University of Texas at Austin, [2]University of Science and Technology of China, [3]AWS Deep Learning

Agenda

- Motivations
- Technical Innovations
- Impressive Findings
- Methodology
- Experiments and observations
- Results out of the Paper

Motivations

Current limitations:

- ➔ With graphs rapidly growing in size and deeper graph neural networks (GNNs) emerging, the training and inference of GNNs become increasingly expensive
- ➔ Graph feature aggregation costs massive computation when the graphs are large and with dense/complicated neighbor connections

Q: To what extent could we co-simplify the input graph and the model, for ultra-efficient GNN inference?

Technical Innovation

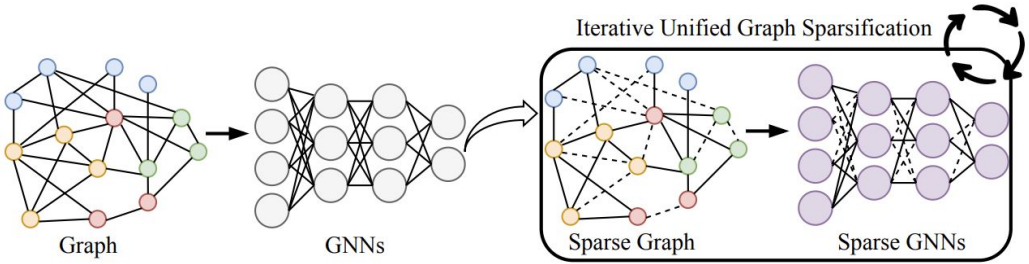
- An end-to-end optimization framework called unified GNN sparsification (UGS) that simultaneously prunes the graph adjacency matrix and the model weights.
- Generalize the popular lottery ticket hypothesis (LTH) to GNNs for the first time.

Impressive Findings

- UGS is widely applicable to simplifying a GNN during training and reducing its inference MACs. Moreover, by iteratively applying UGS, GLTs can be broadly located from for both shallow and deep GNN models, on both small and large-scale graph datasets, with substantially reduced inference costs and unimpaired generalization.
- For node classification, GLTs achieve 20% ~ 98% MACs saving, with up to 5%~ 58.19% sparsity on graphs and 20%~97.75% sparsity on GNN models, at little to no performance degradation. For link prediction, GLTs lead to 48% ~ 97% and 70% MACs saving, coming from up to 22.62% ~ 55.99% sparsity on graphs and and 67.23% ~ 97.19% sparsity on GNN models, again without performance loss.
- Besides from random initializations, GLTs can also be drawn from the initialization via self-supervised pre-training. GLTs can be found to achieve robust performance with even sparser graphs and GNNs.

Methodology - Graph Lottery Tickets

Graph lottery tickets (GLT). Given a GNN $f(\cdot, \boldsymbol{\Theta})$ and a graph $\mathcal{G} = \{\boldsymbol{A}, \boldsymbol{X}\}$, the associated subnetworks of GNN and sub-graph can be defined as $f(\cdot, \boldsymbol{m}_\theta \odot \boldsymbol{\Theta})$ and $\mathcal{G}_s = \{\boldsymbol{m}_g \odot \boldsymbol{A}, \boldsymbol{X}\}$, where $\boldsymbol{m}_g$ and $\boldsymbol{m}_\theta$ are binary masks defined in Section 3.2. If a subnetwork $f(\cdot, \boldsymbol{m}_\theta \odot \boldsymbol{\Theta})$ trained on a sparse graph $\mathcal{G}_s$ has performance matching or surpassing the original GNN trained on the full graph $\mathcal{G}$ in terms of achieved standard testing accuracy, then we define $f(\{\boldsymbol{m}_g \odot \boldsymbol{A}, \boldsymbol{X}\}, \boldsymbol{m}_\theta \odot \boldsymbol{\Theta}_0)$ as a unified *graph lottery tickets* (GLTs), where $\boldsymbol{\Theta}_0$ is the original initialization for GNNs which the found lottery ticket subnetwork is usually trained from.



Methodology - Unified GNN Sparsification

- Algorithm 1 outlines the procedure of UGS (**Unified GNN Sparsification**), and it can be considered as the generalized pruning process for GNNs.
- Objective function:

$$\mathcal{L}_{\text{UGS}} := \mathcal{L}(\{\boldsymbol{m}_g \odot \boldsymbol{A}, \boldsymbol{X}\}, \boldsymbol{m}_\theta \odot \boldsymbol{\Theta}) + \gamma_1 \|\boldsymbol{m}_g\|_1 + \gamma_2 \|\boldsymbol{m}_\theta\|_1,$$

Algorithm 1 Unified GNN Sparsification (UGS)

Input: Graph $\mathcal{G} = \{\boldsymbol{A}, \boldsymbol{X}\}$, GNN $f(\mathcal{G}, \boldsymbol{\Theta}_0)$, GNN's initialization $\boldsymbol{\Theta}_0$, initial masks $\boldsymbol{m}_g^0 = \boldsymbol{A}$, $\boldsymbol{m}_\theta^0 = \mathbf{1} \in \mathbb{R}^{\|\Theta_0\|_0}$, Step size η, λ_g, and λ_θ.

Output: Sparsified masks $\boldsymbol{m}_g$ and $\boldsymbol{m}_\theta$

1: **for** iteration $i = 0, 1, 2, ..., \mathrm{N} - 1$ **do**
2: Forward $f(\cdot, \boldsymbol{m}_\theta^i \odot \boldsymbol{\Theta}_i)$ with $\mathcal{G} = \{\boldsymbol{m}_g^i \odot \boldsymbol{A}, \boldsymbol{X}\}$ to compute the loss $\mathcal{L}_{\text{UGS}}$ in Equation 3.
3: Backpropagate to update $\boldsymbol{\Theta}_{i+1} \leftarrow \boldsymbol{\Theta}_i - \eta \nabla_{\boldsymbol{\Theta}_i} \mathcal{L}_{\text{UGS}}$.
4: Update $\boldsymbol{m}_g^{i+1} \leftarrow \boldsymbol{m}_g^i - \lambda_g \nabla_{\boldsymbol{m}_g^i} \mathcal{L}_{\text{UGS}}$.
5: Update $\boldsymbol{m}_\theta^{i+1} \leftarrow \boldsymbol{m}_\theta^i - \lambda_\theta \nabla_{\boldsymbol{m}_\theta^i} \mathcal{L}_{\text{UGS}}$.
6: **end for**
7: Set $p_g = 5\%$ of the lowest magnitude values in $\boldsymbol{m}_g^{\mathrm{N}}$ to 0 and others to 1, then obtain $\boldsymbol{m}_g$.
8: Set $p_\theta = 20\%$ of the lowest magnitude values in $\boldsymbol{m}_\theta^{\mathrm{N}}$ to 0 and others to 1, then obtain $\boldsymbol{m}_\theta$.

Methodology - Unified GNN Sparsification

- Apply the UGS algorithm to prune both the model and the graph during training.
- Then the GNN weights are rewound to the original initialization.
- We repeat the above two steps iteratively, until reaching the desired sparsity.

Algorithm 2 Iterative UGS to find Graph Lottery Tickets

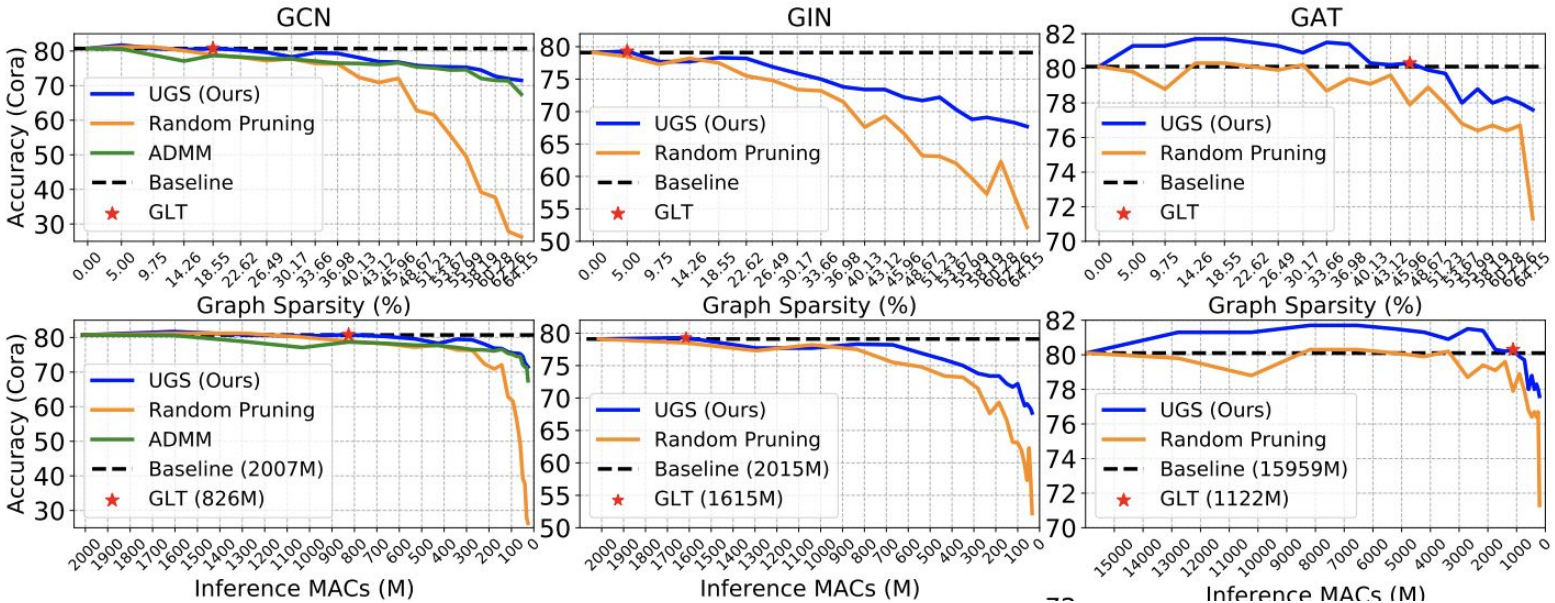
Input: Graph $\mathcal{G} = \{\boldsymbol{A}, \boldsymbol{X}\}$, GNN $f(\mathcal{G}, \boldsymbol{\Theta}_0)$, GNN's initialization $\boldsymbol{\Theta}_0$, pre-defined sparsity levels s_g for graphs and s_θ for GNNs, Initial masks $\boldsymbol{m}_g = \boldsymbol{A}$, $\boldsymbol{m}_\theta = \mathbf{1} \in \mathbb{R}^{\|\Theta_0\|_0}$.

Output: GLT $f(\{\boldsymbol{m}_g \odot \boldsymbol{A}, \boldsymbol{X}\}, \boldsymbol{m}_\theta \odot \boldsymbol{\Theta}_0)$

1: **while** $1 - \frac{\|\boldsymbol{m}_g\|_0}{\|\boldsymbol{A}\|_0} < s_g$ **and** $1 - \frac{\|\boldsymbol{m}_\theta\|_0}{\|\boldsymbol{\Theta}\|_0} < s_\theta$ **do**
2: Sparsify GNN $f(\cdot, \boldsymbol{m}_\theta \odot \boldsymbol{\Theta}_0)$ with $\mathcal{G} = \{\boldsymbol{m}_g \odot \boldsymbol{A}, X\}$ using UGS, as presented in Algorithm 1.
3: Update $\boldsymbol{m}_g$ and $\boldsymbol{m}_\theta$ accordingly.
4: Rewinding GNN's weights to $\boldsymbol{\Theta}_0$
5: Rewinding masks, $\boldsymbol{m}_g = \boldsymbol{m}_g \odot \boldsymbol{A}$
6: **end while**

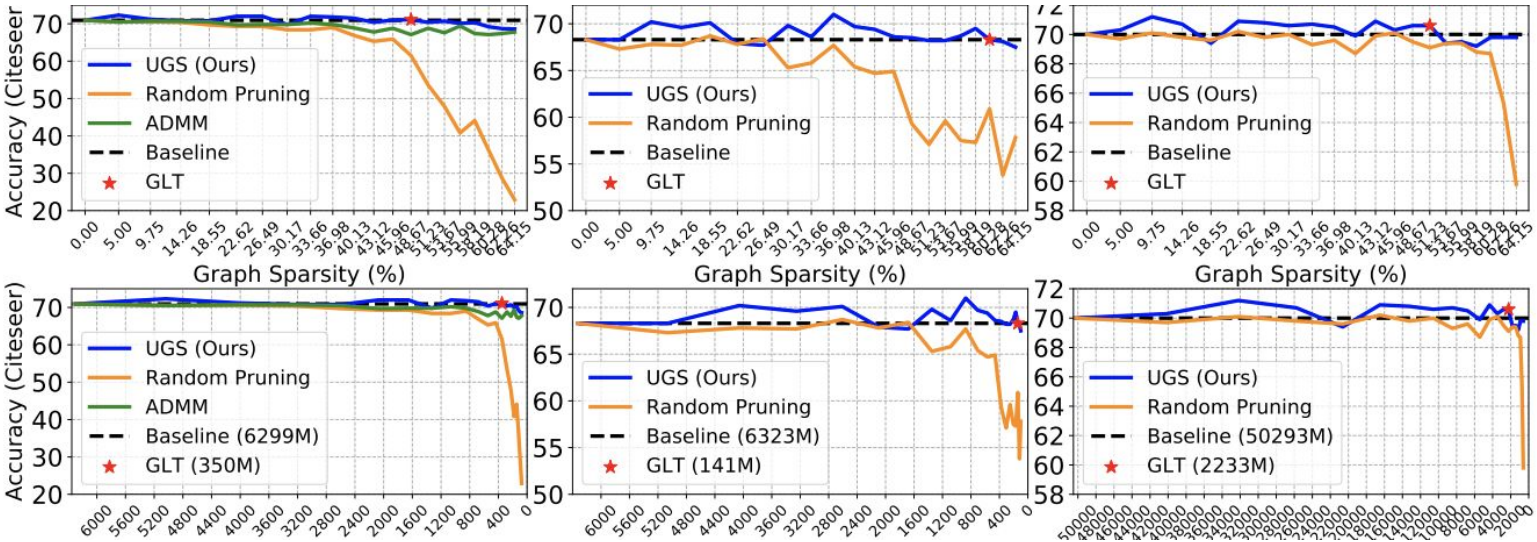
Observation 1 & 2: GLTs exist across different models and datasets, showing superior performance

Node classification



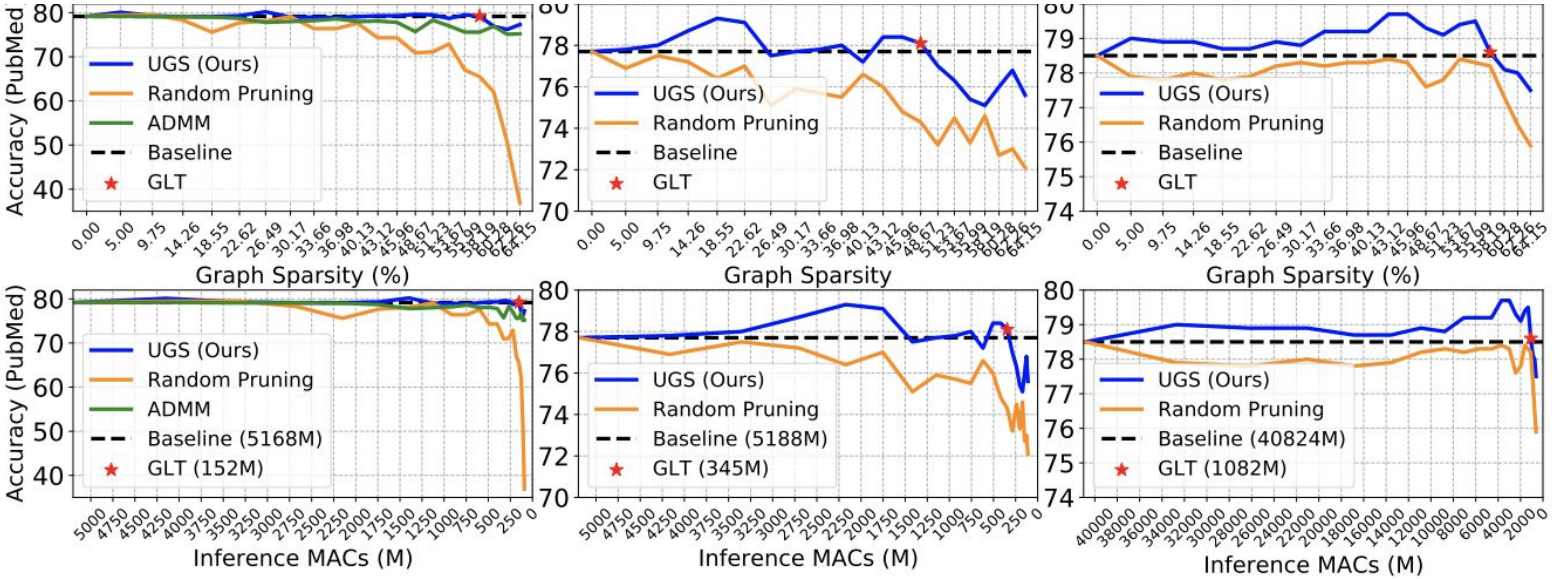
Observation 1 & 2: GLTs exist across different models and datasets, showing superior performance

Node classification



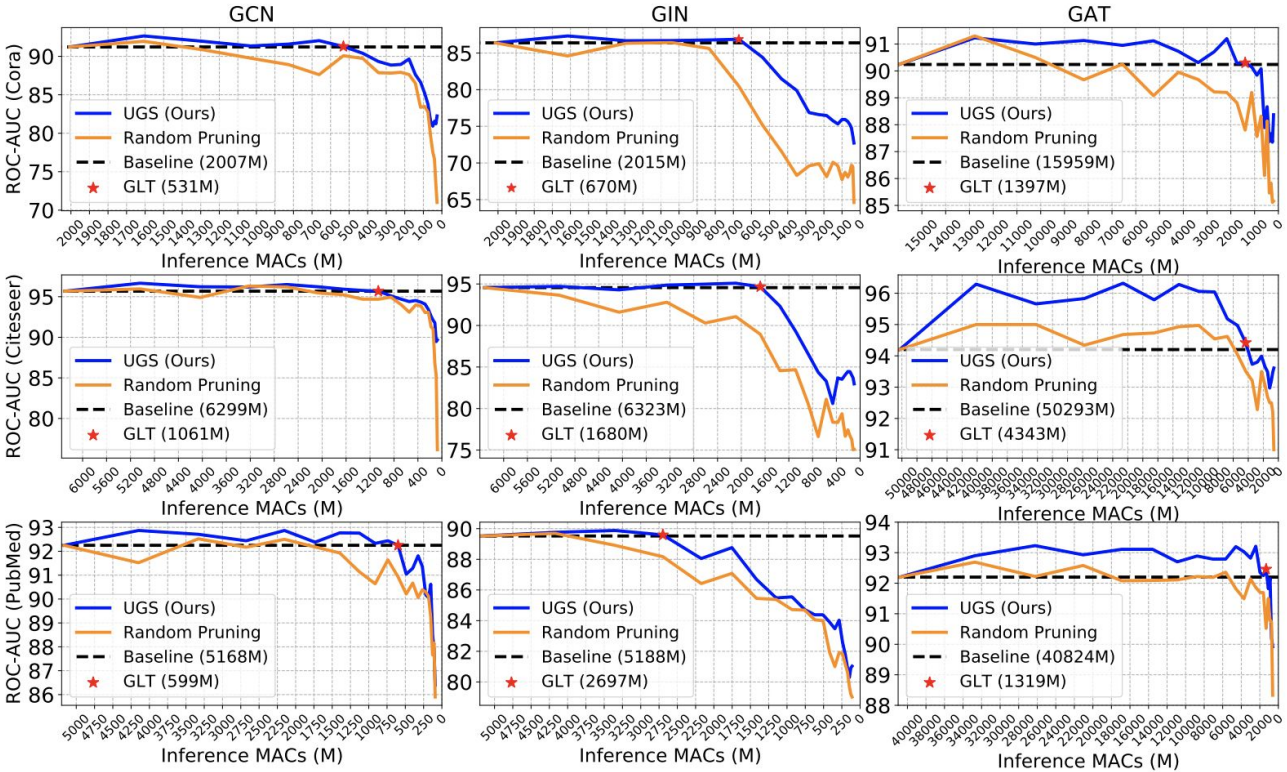
Observation 1 & 2: GLTs exist across different models and datasets, showing superior performance

Node classification



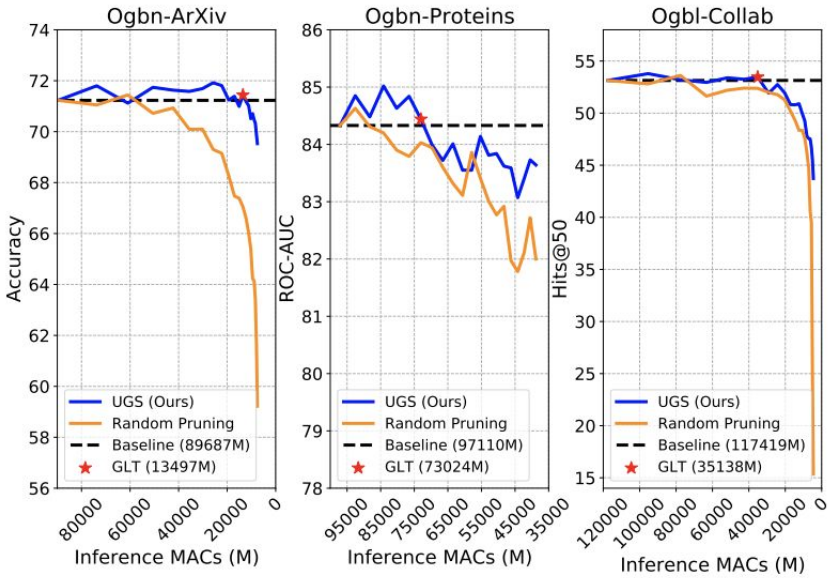
Observation 1 & 2: GLTs exist across different models and datasets, showing superior performance

Link prediction



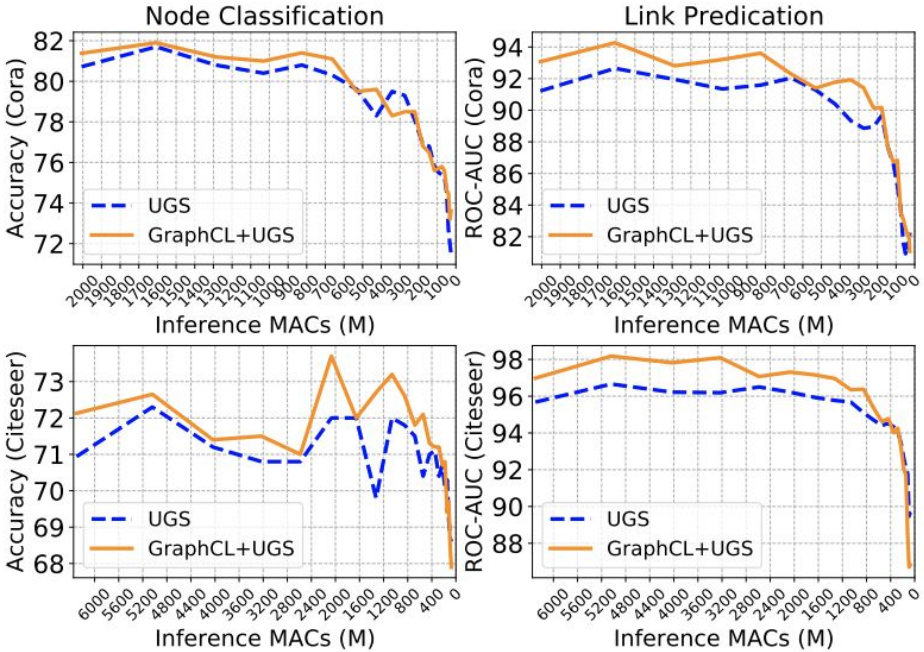
More observations

- Observation 3: GNN-specific and Graph-specific analyses: GAT is more amenable to sparsified graphs; Cora is more sensitive to pruning
- Observation 4: UGS is scalable and GLT exists in deep GCNs on large-scale datasets

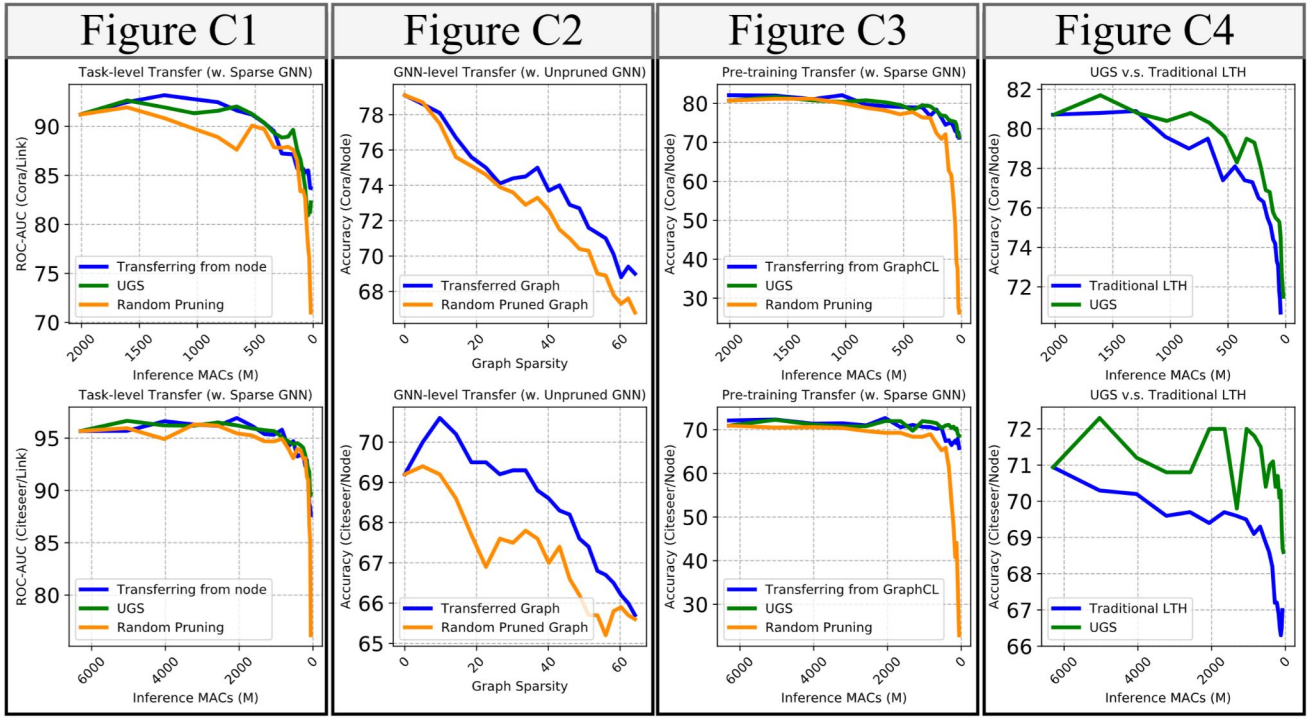


Observation 5: High-quality lottery tickets can be drawn from self-supervised pre-trained models

- UGS with the GraphCL pre-trained initialization consistently presents superior performance at moderate sparsity levels
- GraphCL benefits GLT on multiple downstream tasks
- GLTs with appropriate sparsity levels can even enlarge the performance gain from pre-training



Results out of the Paper (e.g., More Transfer Study)



Q&A

Thanks!