A Unified Lottery Ticket Hypothesis for Graph Neural Networks

Tianlong Chen 1 Yongduo Sui 2 Xuxi Chen 1 Aston Zhang 3 Zhangyang Wang 1

Abstract

With graphs rapidly growing in size and deeper graph neural networks (GNNs) emerging, the training and inference of GNNs become increasingly expensive. Existing network weight pruning algorithms cannot address the main space and computational bottleneck in GNNs, caused by the size and connectivity of the graph. To this end, this paper first presents a unified GNN sparsification (UGS) framework that simultaneously prunes the graph adjacency matrix and the model weights, for effectively accelerating GNN inference on large-scale graphs. Leveraging this new tool, we further generalize the recently popular lottery ticket hypothesis to GNNs for the first time, by defining a graph lottery ticket (GLT) as a pair of core sub-dataset and sparse sub-network, which can be jointly identified from the original GNN and the full dense graph by iteratively applying UGS. Like its counterpart in convolutional neural networks, GLT can be trained in isolation to match the performance of training with the full model and graph, and can be drawn from both randomly initialized and self-supervised pre-trained GNNs. Our proposal has been experimentally verified across various GNN architectures and diverse tasks, on both small-scale graph datasets (Cora, Citeseer and PubMed), and large-scale datasets from the challenging Open Graph Benchmark (OGB). Specifically, for node classification, our found GLTs achieve the same accuracies with 20% ∼ 98% MACs saving on small graphs and 25% ∼ 85% MACs saving on large ones. For link prediction, GLTs lead to 48% ∼ 97% and 70% MACs saving on small and large graph datasets, respectively, without compromising predictive performance. Codes are at https://github.com/VITA-Group/Unified-LTH-GNN.

1. Introduction

Graph Neural Networks (GNNs) (Zhou et al., 2018; Kipf & Welling, 2016; Chen et al., 2019; Veličković et al., 2017) have established state-of-the-art results on various graph-based learning tasks, such as node or link classification (Kipf & Welling, 2016; Veličković et al., 2017; Qu et al., 2019; Verma et al., 2019; Karimi et al., 2019; You et al., 2020b), link prediction (Zhang & Chen, 2018), and graph classification (Ying et al., 2018; Xu et al., 2018; You et al., 2020b). GNNs’ superior performance results from the structure-aware exploitation of graphs. To update the feature of each node, GNNs first aggregate features from neighbor connected nodes, and then transform the aggregated embeddings via (hierarchical) feed-forward propagation.

However, the training and inference of GNNs suffer from the notorious inefficiency, and pose hurdle to GNNs from being scaled up to real-world large-scale graph applications. This hurdle arises from both algorithm and hardware levels. On the algorithm level, GNN models can be thought of a composition of traditional graphs equipped with deep neural network (DNN) algorithms on vertex features. The execution of GNN inference falls into three distinct cat-

Figure 1. Summary of our achieved performance (y-axis) at different graph and GNN sparsity levels (x-axis) on Cora and Citeseer node classification. The size of markers represent the inference MACs (= 1 2 FLOPs) of each sparse GCN on the corresponding sparsified graphs. Black circles (●) indicate the baseline, i.e., unpruned dense GNNs on the full graph. Blue circles (●) are random pruning results. Orange circles (●) represent the performance of a previous graph sparsification approach, i.e., ADMM (Li et al., 2020b). Red stars (★) are established by our method (UGS).
egories with unique computational characteristics: graph traversal, DNN computation, and aggregation. Especially, GNNs broadly follow a recursive neighborhood aggregation (or message passing) scheme, where each node aggregates feature vectors of its multi-hop neighbors to compute its new feature vector. The aggregation phase costs massive computation when the graphs are large and with dense/complicated neighbor connections (Xu et al., 2018). On the hardware level, GNN’s computational structure depends on the often sparse and irregular structure of the graph adjacency matrices. This results in many random memory accesses and limited data reuse, but also requires relatively little computation. As a result, GNNs have much higher inference latency than other neural networks, limiting them to applications where inference can be pre-computed offline (Geng et al., 2020; Yan et al., 2020).

This paper aims at aggressively trimming down the explosive GNN complexity, from the algorithm level. There are two streams of works: simplifying the graph, or simplifying the model. For the first stream, many have explored various sampling-based strategies (Hübler et al., 2008; Chakkeri et al., 2016; Calandriello et al., 2018; Adhikari et al., 2017; Leskovec & Faloutsos, 2006; Voudigari et al., 2016; Eden et al., 2018; Zhao, 2015; Chen et al., 2018a), often combined with mini-batch training algorithms for locally aggregating and updating features. Zheng et al. (2020) investigated graph sparsification, i.e., pruning input graph edges, and learned an extra DNN surrogate. Li et al. (2020b) also addressed graph sparsification by formulating an optimization objective, solved by alternating direction method of multipliers (ADMM) (Bertsekas & Rheinboldt, 1982).

The second stream of efforts were traditionally scarce, since the DNN parts of most GNNs are (comparably) lightly parameterized, despite the recent emergence of increasingly deep GNNs (Li et al., 2019). Although model compression is well studied for other types of DNNs (Cheng et al., 2017), it has not been discussed much for GNNs. One latest work (Tailor et al., 2021) explored the viability of training quantized GNNs, enabling the usage of low precision integer arithmetic during inference. Other forms of well-versed DNN compression techniques, such as model pruning (Han et al., 2016), have not been exploited for GNNs up to our best knowledge. More importantly, no prior discussion was placed on jointly simplifying the input graphs and the models for GNN inference. In view of such, this paper asks: to what extent could we co-simplify the input graph and the model, for ultra-efficient GNN inference?

1.1. Summary of Our Contributions

This paper makes multi-fold contributions to answer the above questions. Unlike pruning convolutional DNNs which are heavily overparameterized, directly pruning the much less parameterized GNN model would have only limited room to gain. Our first technical innovation is to for the first time present an end-to-end optimization framework called unified GNN sparsification (UGS) that simultaneously prunes the graph adjacency matrix and the model weights. UGS makes no assumption to any GNN architecture or graph structure, and can be flexibly applied across various graph-based learning scenarios at scale.

Considering UGS as the generalized pruning for GNNs, our second technical innovation is to generalize the popular lottery ticket hypothesis (LTH) to GNNs for the first time. LTH (Frankle & Carbin, 2018) demonstrates that one can identify highly sparse and independently trainable sub-networks from dense models, by iterative pruning. It was initially observed in convolutional DNNs, and later broadly found in natural language processing (NLP) (Chen et al., 2020b), generative models (Kalibhat et al., 2020), reinforcement learning (Yu et al., 2020) and lifelong learning (Chen et al., 2020b). To meaningfully generalize LTH to GNNs, we define a graph lottery ticket (GLT) as a pair of core sub-dataset and sparse sub-network which can be jointly identified from the full graph and the original GNN model, by iteratively applying UGS. Like its counterpart in convolutional DNNs, a GLT could be trained from its initialization to match the performance of training with the full model and graph, and its inference cost is drastically smaller.

Our proposal has been experimentally verified, across various GNN architectures and diverse tasks, on both small-scale graph datasets (Cora, Citeseer and PubMed), and large-scale datasets from the challenging Open Graph Benchmark (OGB). Our main observations are outlined below:

• UGS is widely applicable to simplifying a GNN during training and reducing its inference MACs (multiply–accumulate operations). Moreover, by iteratively applying UGS, GLTs can be broadly located from 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, we found 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 example in Figure 1, on Cora and Citeseer node classification, our GLTs (●) achieve comparable or sometimes even slightly better performance than the baselines of full models and graphs (●), with only 41.16% and 5.57% MACs, respectively.

• For link prediction, GLTs lead to 48% ~ 97% and 70% MACs saving, coming from up to 22.62% ~ 55.99% sparsity on graphs and 67.23% ~ 97.19% sparsity on GNN models, again without performance loss.

• Our proposed framework can scale up to deep GNN
models (up to 28 layers) on large graphs (e.g., Ogbn-ArXiv and Ogbn-Proteins), without bells and whistles.

- Besides from random initializations, GLTs can also be drawn from the initialization via self-supervised pre-training – an intriguing phenomenon recently just reported for NLP (Chen et al., 2020b) and computer vision models (Chen et al., 2020a). Using a latest GNN pre-training algorithm (You et al., 2020b) for initialization, GLTs can be found to achieve robust performance with even sparser graphs and GNNs.

2. Related Work

Graph Neural Networks. There are mainly three categories of GNNs (Dwivedi et al., 2020): i) extending original convolutional neural networks to the graph regime (Scarselli et al., 2008; Bruna et al., 2013; Kipf & Welling, 2016; Hamilton et al., 2017); ii) introducing anisotropic operations on graphs such as gating and attention (Battaglia et al., 2016; Monti et al., 2017; Velicković et al., 2018), and iii) improving upon limitations of existing models (Xu et al., 2019; Morris et al., 2019; Chen et al., 2019; Murphy et al., 2019).

Among this huge family, Graph Convolutional Networks (GCNs) are widely adopted, which can be categorized as spectral domain based methods (Defferrard et al., 2016; Kipf & Welling, 2016) and spatial domain bases methods (Simonovsky & Komodakis, 2017; Hamilton et al., 2017).

The computational cost and memory usage of GNNs will expeditiously increase with the graph size. The aim of graph sampling or sparsification is to extract a small sub-graph from the original large one, which can remain effective for learning tasks (Zheng et al., 2020; Hamilton et al., 2017) while reducing the cost. Previous works on sampling focus on preserving certain pre-defined graph metrics (Hubler et al., 2008), graph spectrum (Chakeri et al., 2016; Morris et al., 2019; Murphy et al., 2019).

FastGCN (Chen et al., 2018a) introduced a global importance sampling method instead of locally neighbor sampling. VRGCN (Chen et al., 2018b) proposed a control variate based algorithm, but requires all intermediate vertex embeddings to be saved during training. ClusterGCN (Chiang et al., 2019) used clustering to partition subgraphs for training, but often suffers in stability. Zheng et al. (2020); Li et al. (2020b) cast graph sparsification as optimization problems, solved by learning surrogates and ADMM, respectively.

Lottery Ticket Hypothesis (LTH). Since the original LTH (Frankle & Carbin, 2018), a lot of works have explored the prospect of trainable sparse subnetworks in place of the full models without sacrificing performance. Frankle et al. (2019); Renda et al. (2020) introduced the rewinding techniques to scale up LTH. LTH was also adopted in different fields (Evci et al., 2019; Savarese et al., 2020; Liu et al., 2019; You et al., 2020a; Gale et al., 2019; Yu et al., 2020; Kalibhat et al., 2020; Chen et al., 2021b; 2020b,c; 2021a; Ma et al., 2021; Gan et al., 2021).

However, GNN is NOT “yet another” field that can be easily cracked by LTH. That is again due to GNNs having much smaller models, while all the aforementioned LTH works focus on simplifying their redundant models. To our best knowledge, this work is not only the first to generalize LTH to GNNs, but also the first to extend LTH from simplifying models to a new data-model co-simplification prospect.

3. Methodology

3.1. Notations and Formulations

Let \( G = (V, E) \) represent an undirected graph with \( |V| \) nodes and \( |E| \) edges. For \( V = \{v_1, \ldots, v_{|V|}\} \), let \( X \in \mathbb{R}^{|V| \times F} \) denote the node feature matrix of the whole graph, where \( x_i = X[i, :] \) is the \( F \)-dimensional attribute vector of node \( v_i \in V \). As for \( E = \{e_1, \ldots, e_{|E|}\} \), \( e_n = (v_i, v_j) \in E \) means that there exists a connection between node \( v_i \) and \( v_j \). An adjacency matrix \( A \in \mathbb{R}^{|V| \times |V|} \) is defined to describe the overall graph topology, where \( A[i, j] = 1 \) if \( (v_i, v_j) \in E \) else \( A[i, j] = 0 \). For example, the two-layer GNN (Kipf & Welling, 2016) can be defined as follows:

\[
Z = S(\hat{\Delta} \sigma(\hat{A}X \Theta^{(0)}) \Theta^{(1)}),
\]

where \( Z \) is the prediction of GNN \( f(G, \Theta) \). The graph \( G \) can be alternatively denoted as \( \{A, X\}, \Theta = (\Theta^{(0)}, \Theta^{(1)}) \) is the weights of the two-layer GNN, \( S(\cdot) \) represents the softmax function, \( \sigma(\cdot) \) denotes the activation function (e.g., ReLU), \( \hat{A} = \hat{D}^{-\frac{1}{2}} (A + I) \hat{D}^{\frac{1}{2}} \) is normalized by the degree matrix \( \hat{D} \) of \( A + I \). Considering the transductive semi-supervised classification task, the objective function \( \mathcal{L} \) is:

\[
\mathcal{L}(G, \Theta) = -\frac{1}{|V_{\text{label}}|} \sum_{v_i \in V_{\text{label}}} y_i \log(z_i),
\]

where \( \mathcal{L} \) is the cross-entropy loss over all labeled samples \( V_{\text{label}} \subset V \), and \( y_i \) is the annotated label vector of node \( v_i \) for its corresponding prediction \( z_i = Z[i, :] \).

3.2. Unified GNN Sparsification

We present our end-to-end framework, Unified GNN Sparsification (UGS), to simultaneously reduce edges in \( G \) and the parameters in GNNs. Specifically, we introduce two differentiable masks \( m_g \) and \( m_\theta \) for indicating the insignificant connections and weights in the graph and GNNs, respectively. The shapes of \( m_g \) and \( m_\theta \) are identical to those the adjacency matrix \( A \) and the weights \( \Theta \), respectively. Given \( A, \Theta, m_g \) and \( m_\theta \) are co-optimized from end to end, under the following objective:

\[
\mathcal{L}_{\text{UGS}} := \mathcal{L}(m_g \odot A, X; m_\theta \odot \Theta) + \gamma_1 \|m_g\|_1 + \gamma_2 \|m_\theta\|_1,
\]
where ⊙ is the element-wise product, γ₁ and γ₂ are the hyparameters to control ℓ₁ sparsity regularizers of mg and mθ respectively. After the training is done, we set the lowest-magnitude elements in mg and mθ to zero, w.r.t. pre-defined ratios p₀ and p₁. Then, the two sparse masks are applied to prune A and Θ, leading to the final sparse graph and model. Alg. 1 outlines the procedure of UGS, and it can be considered as the generalized pruning for GNNs.

### 3.3. Graph Lottery Tickets

**Graph lottery tickets (GLT).** Given a GNN f(·, Θ) and a graph G = {A, X}, the associated subnetworks of GNN and sub-graph can be defined as f(·, mg ⊙ Θ) and Gs = {mg ⊙ A, X}, where mg and mθ are binary masks defined in Section 3.2. If a subnetwork f(·, mg ⊙ Θ) trained on a sparse graph Gs, has performance matching or surpassing the original GNN trained on the full graph G in terms of achieved standard testing accuracy, then we define f(·, mg ⊙ A, X, Θ₀) as a unified graph lottery tickets (GLTs), where Θ₀ is the original initialization for GNNs which the found lottery ticket subnetwork is usually trained from.

Unlike previous LTH literature (Frankle & Carbin, 2018), our identified GLT will consist of three elements: i) a sparse graph G, ii) an associated subnetwork of GNN s, and iii) the model weight’s initialization Θ₀. We repeat the above two steps iteratively, until reaching the desired sparsity s and s₀ for the graph and GNN, respectively.

#### Complexity analysis of GLTs.

The inference time complexity of GLTs is O(L × ||mg ⊙ A||₀ × F + L × ||mθ||₀ × |V| × F²), where L is the number of layers, ||mg ⊙ A||₀ is the number of remaining edges in the sparse graph, F is the dimension of node features, |V| is the number of nodes. The memory complexity is O(L × |V| × F + L × ||mθ||₀ × F²). In our implementation, pruned edges will be removed from E, and would not participate in the next round’s computation.

### 4. Experiments

In this section, extensive experiments are reported to validate the effectiveness of UGS and the existence of GLTs across diverse graphs and GNN models. Our subjects include small- and medium-scale graphs with two-layer Graph Neural Networks.
Convolutional Network (GCN) (Kipf & Welling, 2016), Graph Isomorphism Network (GIN) (Xu et al., 2018), and Graph Attention Network (GAT) (Veličković et al., 2017) in Section 4.2; as well as large-scale graphs with 28-layer deep ResGCNs (Li et al., 2020a) in Section 4.2. Besides, in Section 4.3, we investigate GLTs under the self-supervised pre-training (You et al., 2020b). Ablation studies and visualizations are provided in Section 4.4 and 4.5.

Figure 3. Node classification performance over achieved graph sparsity levels or inference MACs of GCN, GIN, and GAT on Cora, Citeseer, and PubMed datasets, respectively. Red stars (*) indicate the located GLTs, which reach comparable performance with high sparsity and low inference MACs. Dash lines represent the baseline performance of full GNNs on full graphs. More results over GNN sparsity are provided in Appendix A2.1.
Datasets We use popular semi-supervised graph datasets: Cora, Citeseer and PubMed (Kipf & Welling, 2016), for both node classification and link prediction tasks. For experiments on large-scale graphs, we use the Open Graph Benchmark (OGB) (Hu et al., 2020), such as Ogbn-Arxiv, Ogbn-Proteins, and Ogbn-Collab. More datasets statistics are summarized in Table 1. Other details such as the datasets’ train-val-test splits are included in Appendix A1.

Training and Inference Details Our evaluation metrics are shown in Table 1, following Kipf & Welling (2016); Hu et al. (2020); Mavromatis & Karypis (2020). More detailed configurations such as learning rate, training iterations, and hyperparameters in UGS, are referred to Appendix A1.

4.1. The Existence of Graph Lottery Ticket

We first examine whether unified graph lottery tickets exist and can be located by UGS. Results of GCN/GIN/GAT on Cora/Citesser/PubMed for node classification and link prediction are collected in Figures 3 and 4, respectively. Note that each point in the figures denotes the achieved performance with respect to a certain graph sparsity, GNN sparsity, and inference MACs. However, due to the limited space, we only include one or two of these three sparsity indicators in the main text, and the rest can be found in Appendix A2. We list the following Observations.

Obs. 1. GLTs broadly exist with substantial MACs saving. Graph lottery tickets at a range of graph sparsity from 5% to 58.19% without performance deterioration, can be identified across GCN, GIN and GAT on Cora, Citeseer, and PubMed datasets for both node classification and link prediction tasks. Such GLTs significantly reduce 59% ~ 97%, 20% ~ 98%, 91% ~ 97% inference MACs for GCN, GIN and GAT across all datasets.

Obs. 2. UGS is flexible and consistently shows superior performance. UGS consistently surpasses random pruning by substantial performance margins across all datasets and GNNs, which validates the effectiveness of our proposal. The previous state-of-the-art method, i.e., ADMM (Li et al., 2020b), achieves a competitive performance to UGS at moderate graph sparsity levels, and performs 3 ~ 4% worse than UGS when graphs are heavily pruned. Note that the ADMM approach by Li et al. (2020b) is
only applicable when two conditions are met: i) graphs are stored via adjacency matrices—however, that is not practical for large graphs (Hu et al., 2020); ii) aggregating features with respect to adjacency matrices—however, recent designs of GNNs (e.g., GIN and GAT) commonly use the much more computation efficient approach of synchronous/asynchronous message passing (Gilmer et al., 2017; Busch et al., 2020). On the contrary, our proposed UGS is flexible enough and free of these limitations.

Obs. 3. GNN-specific and Graph-specific analyses: GAT is more amenable to sparsified graphs; Cora is more sensitive to pruning. As demonstrated in Figures 3 and 4, compared to GCN and GIN, GLTs in GAT can be found at higher sparsity levels; meanwhile randomly pruned graphs and GAT can still reach satisfied performance and maintain higher accuracies on severely sparsified graphs.

One possible explanation is that attention-based aggregation is capable of re-identifying important connections in pruned graphs which makes GAT be more amenable to sparsification. Compared the sparsity of located GLTs (i.e., the position of red stars (★)) across three graph datasets, we find that Cora is the most sensitive graph to pruning and PubMed is more robust to be sparsified.

4.2. Scale Up Graph Lottery Tickets

To scale up graph lottery tickets, we further conduct experiments on 28-layer deep ResGCNs (Li et al., 2020a) on large-scale datasets that have more than millions of connections, like Ogbn-Arxiv and Ogbn-Proteins for node classification, Ogbl-Collab for link prediction in Table 1. We summarize our observations and derive insights below.

Figure 5. Node classification and link prediction performance of 28-layer deep ResGCNs on large-scale graph datasets.

Obs. 5. Denser graphs (e.g., Ogbn-Proteins) are more resilient to sparsification. As shown in Figure 5, comparing the node classification results on Ogbn-Arxiv (Ave. degree: 13.77) and Ogbn-Proteins (Ave. degree: 597.00), Ogbn-Proteins has a negligible performance gap between UGS and random pruning, even on heavily pruned graphs. Since nodes with high degrees in denser graphs have less chance to be totally isolated during pruning, it may contribute to more robustness to sparsification. Similar observations can be drawn from the comparison between PubMed and other two small graphs in Figure 3 and 4.

4.3. Graph Lottery Ticket from Pre-training

High-quality lottery tickets can be drawn from self-supervised pre-trained models, as recently found in both NLP and computer vision fields (Chen et al., 2020b; a). In the GLT case, we also assess the impact of replacing random initialization with self-supervised graph pre-training, i.e., GraphCL (You et al., 2020b), on transductive semi-supervised node classification and link prediction.

From Figure 6 and A12, we gain a few interesting observations. First, UGS with the GraphCL pre-trained initialization consistently presents superior performance at moderate sparsity levels (≤ 40% graph sparsity ≲ ≤ 85% MACs saving). While the two settings perform similar at extreme sparsity, it indicates that for excessively pruned graphs, the initialization is no longer the performance bottleneck; Second, GraphCL benefits GLT on multiple downstream tasks including node classification and link prediction; Third, especially on the transductive semi-supervised setup, GLTs with appropriate sparsity levels can even enlarge the performance gain from pre-training, for example, see GLT on Citeseer with 22.62% graph sparsity and 2068M inference MACs.
A Unified Lottery Ticket Hypothesis for Graph Neural Networks

4.4. Ablation Study

Pruning ratio $p_g$ and $p_\theta$. We extensively investigate the pruning ratios $p_g, p_\theta$ in UGS for graph and GNN sparsification. As shown in Figure 7, with a fixed $p_\theta = 20\%$, only the setting of $p_g = 5\%$ can identify the GLT, and it performs close to $p_g = 10\%$ at higher sparsity levels (e.g., $\geq 25\%$). Aggressively pruning the graph’s connections in each round of iterative UGS, e.g., $p_g = 20\%$, leads to substantially degraded accuracies, especially for large sparsities. On the other hand, with a fixed $p_g = 20\%$, all the settings of $p_\theta = 10\%, 20\%, 40\%$ show similar performance, and even higher pruning ratios produce slight better results. It again verifies that the key bottleneck in pruning GNNs mainly lies in the sparsification of graphs. In summary, we adopt $p_g = 5\%$ and $p_\theta = 20\%$ (follow previous LTH works (Frankle & Carbin, 2018)) for all the experiments.

![Figure 7. Ablation studies of pruning ratios for the graph and GNN sparsification by UGS, i.e., $p_g$, $p_\theta$. Each curve records the achieved performance of 20 rounds iterative UGS. GCN on Cora is adopted here. The embedded sub-graph is a zoom-in of the red box region.](image)

<table>
<thead>
<tr>
<th>Settings</th>
<th>(Graph Sparsity, GNN Sparsity)%(s, s%)</th>
<th>(18.55,59.04)</th>
<th>(22.62,67.23)</th>
<th>(36.98,86.58)</th>
<th>(55.99,97.19)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random GLT</td>
<td>79.70</td>
<td>78.50</td>
<td>75.70</td>
<td>63.70</td>
<td></td>
</tr>
<tr>
<td>GLT</td>
<td>80.80</td>
<td>80.30</td>
<td>79.30</td>
<td>75.30</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Performance comparisons of Random GLT versus GLT from GCN on Cora, at several sparsity levels.

Random graph lottery tickets. Randomly re-initializing located sparse models, i.e., random lottery tickets, usually serves as a necessary baseline for validating the effectiveness of rewinding processes (Frankle & Carbin, 2018). In Table 2, we compare GLT to Random GLT, the latter by randomly re-initializing GNN’s weights and learnable masks, and GLT shows apparently superior performance, consistent with previous observations (Frankle & Carbin, 2018).

4.5. Visualization and Analysis

In this section, we visualize the sparsified graphs in GLTs from UGS in Figure 8, and further measure the graph properties shown in Table A5, including clustering coefficient (Luce & Perry, 1949), as well as node and edge betweenness centrality (Freeman, 1977). Specifically, clustering coefficient measures the proportion of edges between the nodes within a given node’s neighborhood; node and edge betweenness centrality show the degree of central a vertex or an edge is in the graph (Narayanan, 2005). Reported numbers in Table A5 are averaged over all the nodes.

Both Figure 8 and Table A5 show that sparse graphs obtained from UGS seem to maintain more “critical” vertices which used to have much denser connections. It may provide possible insights on what GLTs prefer to preserve.

5. Conclusion and Discussion

In this paper, we first propose unified GNN sparsification to generalize the notion or pruning in GNNs. We further establish the LTH for GNNs, by leveraging UGS and considering a novel joint data-model lottery ticket. The new unified LTH for GNNs generalizes across various GNN architectures, learning tasks, datasets, and even initialization ways. In general, we find GLT to tremendously trim down the inference MACs, without sacrificing task performance.

It remains open how much we could translate GLT’s high sparsity into practical acceleration and energy-saving benefits. Most DNN accelerators are optimized for dense and regular computation, making edge-based operations hard to implement efficiently. To our best knowledge, the hardware acceleration research on GNNs just starts to gain interests (Auten et al., 2020; Abadal et al., 2020; Geng et al., 2020; Wang et al., 2020; Kinningham et al., 2020). We expect GLT to be implemented using sparse-dense matrix multiplication (SpMM) operations from highly optimized sparse matrix libraries, such as Intel MKL (Wang et al., 2014) or cuSPARSE (Naumov et al., 2010).

---

1NetworkX (https://networkx.org) is used for our analyses.
References


A Unified Lottery Ticket Hypothesis for Graph Neural Networks


A Unified Lottery Ticket Hypothesis for Graph Neural Networks


