Fast Track to Winning Tickets: Repowering One-Shot Pruning for Graph Neural Networks

Yanwei Yue*, Guibin Zhang*, Haoran Yang, Dawei Cheng†

Department of Computer Science and Technology, Tongji University
{2053965, bin2003, 2010498, dcheng}@tongji.edu.cn

Abstract

Graph Neural Networks (GNNs) demonstrate superior performance in various graph learning tasks, yet their wider real-world application is hindered by the computational overhead when applied to large-scale graphs. To address the issue, the Graph Lottery Hypothesis (GLT) has been proposed, advocating the identification of subgraphs and subnetworks, *i.e.*, winning tickets, without compromising performance. The effectiveness of current GLT methods largely stems from the use of iterative magnitude pruning (IMP), which offers higher stability and better performance than one-shot pruning. However, identifying GLTs is highly computationally expensive, due to the iterative pruning and retraining required by IMP. In this paper, we reevaluate the correlation between one-shot pruning and IMP: while one-shot tickets are suboptimal compared to IMP, they offer a *fast track* to tickets with a stronger performance. We introduce a one-shot pruning and denoising framework to validate the efficacy of the *fast track*. Compared to current IMP-based GLT methods, our framework achieves a double-win situation of graph lottery tickets with **higher sparsity** and **faster speeds**. Through extensive experiments across 4 backbones and 6 datasets, our method demonstrates a $1.32\%-45.62\%$ improvement in weight sparsity and a $7.49\% - 22.71\%$ increase in graph sparsity, along with a $1.7 - 44\times$ speedup over IMP-based methods and $95.3\% - 98.6\%$ MAC savings.

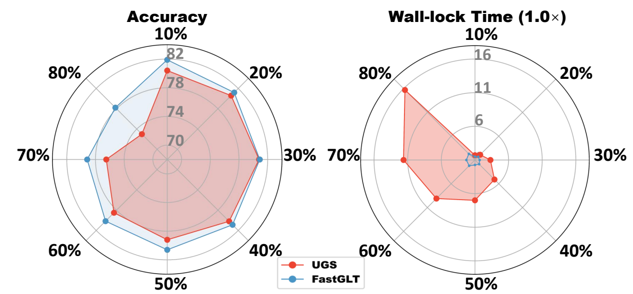

Figure 1: (***Left***) Accuracy (↑) of UGS and `FastGLT` on Cora+GAT, with fixed weight sparsity $s_\theta = 90\%$ and graph sparsity $s_g \in \{10\%, 20\%, \cdots, 80\%\}$ (***Right***) Relative wall-clock time (↓) compared to a single baseline training for searching GLTs. Note that `FastGLT` requires far less wall-lock time to obtain subnetwork/subgraph with better performance than multiple rounds of IMP employed in UGS.

1 Introduction

Graph Neural Networks (GNN) (Kipf and Welling 2016; Hamilton, Ying, and Leskovec 2017) have recently become the predominant approaches for various graph-related learning challenges, including node classification (Velickovic et al. 2017; Cheng et al. 2023; Wang et al. 2023a, 2024), link prediction (Zhang and Chen 2018, 2019), and graph classification (Ying et al. 2018; Zhang et al. 2018; Fang et al. 2024). Nonetheless, the significant computational challenges primarily arise from the over-parameterized GNN weights that are equipped with dense connections, as well as from the large-scale graph samples as input. These factors impede efficient feature aggregation during the training and inference processes of GNNs (Jin et al. 2021; Zhang et al. 2024b). Worse still, these intrinsic limitations curtail the application of GNNs in large-scale scenarios, particularly under resource-restricted conditions.

Prudently reflecting on prior research, the majority of efforts to tackle this inefficiency have concentrated on either (1) *reducing the network's parameters* or (2) *sparsifying the input graph* (Zhang et al. 2024b; Chen, Ma, and Xiao 2018; Zhang et al. 2024a,c). The first class typically employs methods like quantization (Tailor, Fernandez-Marques, and Lane 2020), pruning (Zhou et al. 2021), and distillation (Chen et al. 2020) to streamline GNN parameters. The second category involves leveraging graph sampling or sparsification techniques to reduce the computational demands caused by dense graphs (Chen, Ma, and Xiao 2018; Eden et al. 2018; Li et al. 2020b). Recently, the Graph Lottery Ticket hypothesis (GLT) (Chen et al. 2021b) takes the first step to unify the above two research lines. Briefly put, GLT aims to identify a *graph lottery ticket*, *i.e.*, a combination of *a core subgraph and a sparse subnetwork* with admirable performance for accelerating GNN training and inference process. Such a hypothesis is inspired by the Lottery Ticket Hypothesis (LTH) (Frankle and Carbin 2018), which posits that sparse but performant subnetworks exist in a dense network with random initialization, like *winning tickets* in

*These authors contributed equally.
†Corresponding Author

a lottery pool. Considering its exceptional potential, subsequent research has developed GLT from both theoretical (Bai et al. 2022; Wang et al. 2023b) and algorithmic perspective (Zhang et al. 2024b; Wang et al. 2023d, 2022; You et al. 2022).

As one of the most well-established graph and weight pruning approaches, the success of GLT is attributed to the application of IMP, whose fundamental concept is to involve iteratively pruning and retraining the model, methodically eliminating a small percentage of the remaining weights in each cycle, and persisting until the target pruning ratio is achieved. Unfortunately, **the computational cost of IMP becomes excessively high as the targeted pruning ratio rises and pruned graph volume grows (You et al. 2022; Zhang et al. 2021),** as shown in Fig. 1 (*Right*). To decrease computational expenses, several efficient one-shot magnitude pruning (OMP) methods have been introduced (Lee, Ajanthan, and Torr 2018; Wang, Zhang, and Grosse 2020; Ma et al. 2021), which directly prune the model to the desired sparsity level. However, they (1) typically exhibit a notable performance degradation compared to IMP (Ma et al. 2021; Frankle et al. 2020) and (2) mainly focus on weight pruning, and their performance in the context of joint pruning (graph/GNN) within the GLT context remains unexplored and unknown.

In this work, we take the first step to explore the feasibility of utilizing one-shot pruning in place of IMP within the GLT context, aiming to break the persistent challenge that the performance of one-shot pruning has been inferior to IMP. Towards this end, we introduce a One-shot Pruning and Denoising Framework toward Fast Track Graph Lottery Tickets (termed `FastGLT`). Technically, `FastGLT` initially obtains tickets at a sparsity level close to the target through one-shot pruning, followed by denoising these tickets based on gradient and degree metrics to achieve performance comparable to traditional GLT derived from IMP. Our rationale for this approach stems from a straightforward motivation: Although subnetworks/subgraphs revealed by one-shot pruning are less optimal than those from IMP, the gap between these suboptimal tickets and IMP's winning tickets is minimal and exhibits consistent patterns. Therefore, these one-shot tickets represent a *fast track* to winning tickets. By denoising them, we can swiftly locate GLTs with significantly lower computational costs than those with IMP (shown in Fig. 1). Our contributions can be summarized as follows:

- We re-evaluate one-shot pruning within the context of graph lottery tickets, hypothesizing and empirically validating a *fast track* whereby one-shot tickets directly lead to high-performing winning tickets.
- We introduce a one-shot pruning and denoising framework (`FastGLT`) for efficiently identifying GLTs. `FastGLT` forgoes the expensive IMP steps in traditional ones, leveraging one-shot tickets as a fast track toward winning tickets accompanied by performance that is in no way inferior to that of IMP.
- Extensive experiments on 6 datasets and 4 GNN architectures show that (i) `FastGLT` achieves significant improvements in both weight sparsity ($5.82\% - 25.48\% \uparrow$) and graph sparsity ($3.65\% - 17.48\% \uparrow$) compared to current state-of-the-art GLT methods (Chen et al. 2021b; Wang et al. 2023d), and (ii) `FastGLT` demonstrates substantial efficiency, achieving a 1.7-44$\times$ speedup over IMP-based GLTs and $95.3\% - 98.6\%$ MAC savings.

2 Preliminary & Motivation

2.1 Notations

We consider an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, with $\mathcal{V}$ as the node set and $\mathcal{E}$ the edge set of $\mathcal{G}$. The feature matrix of $\mathcal{G}$ is represented as $\mathbf{X} \in \mathbb{R}^{N\times F}$, where $N = |\mathcal{V}|$ signifies the total number of nodes in the graph. The feature vector for each node $v_i \in \mathcal{V}$, with F dimensions, is denoted by $x_i = \mathbf{X}[i,\cdot]$. An adjacency matrix $\mathbf{A} \in \{0,1\}^{N\times N}$ is utilized to depict the inter-node connectivity, where $\mathbf{A}[i,j] = 1$ indicates an edge $e_{ij} \in \mathcal{E}$, and 0 otherwise. Let $f(\cdot;\mathbf{\Theta})$ represent a GNN model with $\mathbf{\Theta}$ as its parameters. For instance, a two-layer GCN is formulated as:

$$\mathbf{Z} = f(\{\mathbf{A},\mathbf{X}\};\mathbf{\Theta}) = \text{Softmax}(\hat{A}\sigma(\hat{\mathbf{A}}\mathbf{X}\mathbf{\Theta}^{(0)})\mathbf{\Theta}^{(1)}), \quad (1)$$

where $\mathbf{Z}$ denotes the output, $\hat{\mathbf{A}} = \hat{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I}_n)\hat{\mathbf{D}}^{-\frac{1}{2}}$ is the normalized adjacency matrix, $\tilde{\mathbf{A}} = \mathbf{A}+\mathbf{I}_n$, $\hat{\mathbf{D}}$ is the degree matrix of $\hat{\mathbf{A}}$, $\sigma(\cdot)$ is an activation function, and $\mathbf{\Theta}^{(k)}$ is the weight matrix at the k-th layer.

2.2 Graph Lottery Ticket

Given an input graph $\mathcal{G}$ and a GNN model $f(\cdot;\mathbf{\Theta})$, let $\mathcal{G}_{\text{sub}} = \{\mathbf{A}\odot\mathbf{M}_g, \mathbf{X}\}$ be a subgraph of $\mathcal{G}$ and $f_{\text{sub}}(\cdot;\mathbf{\Theta}\odot\mathbf{M}_\theta)$ be a subnetwork of $f(\cdot;\mathbf{\Theta})$. Here, $\mathbf{M}_g$ and $\mathbf{M}_\theta$ are binary mask matrices for the adjacency matrix and model weights, respectively. Additionally, we can define the graph sparsity (GS) s_g and weight sparsity (WS) s_θ as follows:

$$s_g = 1 - \frac{||\mathbf{M}_g||_0}{||\mathbf{A}||_0}, \quad s_\theta = 1 - \frac{||\mathbf{M}_\theta||_0}{||\mathbf{\Theta}||_0}, \quad (2)$$

where the $||\cdot||_0$ denotes the ℓ_0 norm that counts the number of non-zero elements. The graph lottery ticket (GLT) is defined with $\mathcal{G}_{\text{sub}}$ and f_{sub} as follows:

Definition 1 (Graph Lottery Ticket). *Let $\mathcal{G}$ represent an input graph, and let $f(\cdot;\mathbf{\Theta})$ denote a GNN with model parameters initialized at $\mathbf{\Theta}_0$. We define a graph lottery ticket as the pair $(\mathcal{G}_{sub}, f_{sub})$, where $\mathcal{G}_{sub}$ is a sparsified version of $\mathcal{G}$ and f_{sub} corresponds to a sparsified model. This ticket satisfies the condition that, when trained in isolation, the performance metric $\varphi(f_{sub}(\mathcal{G}_{sub};\mathbf{\Theta}_0))$ is at least $\varphi(f(\mathcal{G};\mathbf{\Theta}_0))$, where φ denotes the test accuracy.*

We define the *extreme graph/weight sparsity* of a GLT method as the maximum graph/weight sparsity where it successfully identifies GLTs.

2.3 Motivation

Comparison & Visualization. We define the graph mask $\mathbf{M}_g$ produced by UGS (Chen et al. 2021b) through iterative magnitude pruning as *IMP-based masks*, those pruned by UGS directly to the target sparsity as *one-shot masks*, and masks from random pruning as *random masks*. Fig. 2 (*Left*)

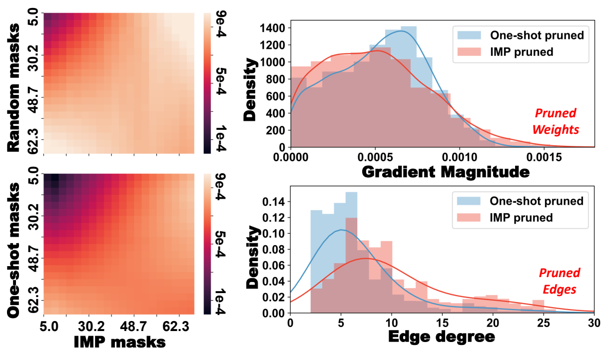

Figure 2: (***Left***) Hamming distance between masks generated by IMP, one-shot, and random sparsification methods on Cora with various graph sparsity levels $s_g \in \{5.0\%, 9.8\%, \ldots, 64.2\%\}$. Notably, as sparsity increases, the distance between random and IMP masks rapidly grows, whereas one-shot masks retain greater similarity to IMP masks. (***Right***) Comparison of gradient magnitude/edge degree for weights/edges pruned by IMP or one-shot pruning.

illustrates the Hamming distance (You et al. 2022) between these masks at different sparsity levels, which can effectively reflect their differences. It is noticeable that the disparity between random and IMP masks (termed structural noise) snowballs with increasing sparsity. In contrast, the noise between one-shot and IMP masks consistently remains minimal. This prompts us to consider: *What characteristics define these structural noises?*

Emperical Validation. Further, we employed gradients (Evci et al. 2020; Lee, Ajanthan, and Torr 2018) and edge degrees[1] (Wang et al. 2022) to visualize differences in $\mathbf{M}_\theta$ and $\mathbf{M}_g$ between one-shot and IMP tickets, as shown in Fig. 2 (*Right*). Observations reveal that (1) weights pruned by IMP exhibit generally smaller gradients than those pruned one-shot, (2) degrees of edges pruned by IMP are significantly lower than those pruned one-shot. We draw an intuitive conclusion that, compared to IMP tickets, one-shot pruning's suboptimal performance stems from mistakenly pruning a minority of weights with higher gradients and edges with lower edge degrees. A natural question arises: *can we enhance one-shot tickets' performance by denoising structural noise using gradient- and degree-based metrics?* This will be empirically validated in the following sections.

3 Methodology

Fig. 3 illustrates the comparison between our `FastGLT` and IMP-based GLT methods like UGS and WD-GLT (Hui et al. 2023). Fig. 3 (*Up*) depicts how traditional methods iteratively prune and retrain through k iterations (each taking E epochs) to achieve a GLT at target sparsity $S\%$. Conversely, Fig. 3 (*Down*) shows that `FastGLT` employs a one-shot pruning fast track to closely approach the target sparsity, followed by a gradual denoising process to fine-tune one-shot tickets towards better performance. This approach requires a total computational budget of $E + D$ epochs, significantly less than IMP's $k \times E$. In subsequent subsections, we outline in Sec. 3.1 how `FastGLT` acquires one-shot tickets, and then elaborate on how the gradual denoising mechanism refines these tickets in Sec. 3.2.

3.1 One-shot Pruning Tickets

As depicted above, we assign two trainable masks $\boldsymbol{m}_g$ and $\boldsymbol{m}_\theta$ on input graph $\mathcal{G}$ and GNN model $f(\cdot;\boldsymbol{\Theta})$ (initialized with $\boldsymbol{\Theta}_0$). Firstly, we co-optimize $\mathbf{A}$, $\boldsymbol{\Theta}$, $\boldsymbol{m}_g$, and $\boldsymbol{m}_\theta$ in an end-to-end manner using the following objective function:

$$\mathcal{L}_{\text{os}} = \mathcal{L}\left(f_{sub}\left(\{\boldsymbol{m}_g \odot \mathbf{A}, \mathbf{X}\}, \boldsymbol{m}_\theta \odot \boldsymbol{\Theta}\right); y\right), \tag{3}$$

where $\mathcal{L}$ denotes the task-itrelevant loss function (**i.e.**, cross-entropy loss), $\odot$ denotes element-wise multiplication and y denotes the node labels. Different from UGS (Chen et al. 2021b), we do not impose ℓ_1 regularization on $\boldsymbol{m}_g$ and $\boldsymbol{m}_\theta$ to reduce the usage of hyperparameters. Upon completing the training, we select the optimal masks, *i.e.*, $\boldsymbol{m}_g$ and $\boldsymbol{m}_\theta$ from the epoch with the highest validation score, denoted as $\boldsymbol{m}_g^*$ and $\boldsymbol{m}_\theta^*$, for pruning.

Given the target graph sparsity s_g^{tgt} and weight sparsity s_θ^{tgt}, we avoid pruning directly to these target sparsities. This is due to the potential performance collapse (Hui et al. 2023) when targeting extremely high sparsity (*e.g.*, 99% weight sparsity or 80% graph sparsity), which can render the model untrainable. Instead, we utilize an exponential decay function $\Psi(s) = s - \alpha s^\beta$ to pre-calculate an intermediate sparsity s_g^{inm} and s_θ^{inm} based on the target sparsities, and α and β are coefficients to adjust the output. We then zero the lowest-magnitude elements in $\boldsymbol{m}_g^*$ and $\boldsymbol{m}_\theta^*$ w.r.t. s_g^{inm} and s_θ^{inm} in a unified manner, as outlined below:

$$\mathbf{M}^{\circledcirc} = \mathbb{1}\left[\boldsymbol{m}^*\right] \odot \mathbb{1}\left[|\boldsymbol{m}^*| > \text{Thresholding}(\boldsymbol{m}^*, \Psi(s^{\text{tgt}}))\right], \tag{4}$$

where $\mathbf{M}^{\circledcirc} \in \{0,1\}^{|\boldsymbol{m}^*|}$ is either the one-shot graph mask $\mathbf{M}_g^{\circledcirc}$ or weight mask $\mathbf{M}_\theta^{\circledcirc}$, $\mathbb{1}[\cdot]$ is a binary indicator, and $\text{Thresholding}(m, s)$ denotes caculating the global threshold value at top s by sorting m in descending order.

3.2 Gradual Denoising Mechanism

As stated in (Wang et al. 2023d), traditional GLT methods irreversibly exclude elements pruned in a given iteration from subsequent considerations, leading to information loss in the pruned subgraph/subnetwork. This aligns with our assertion: one-shot masks may contain noisy and ineffective elements, while pruned parts could hold valuable structures. Towards this end, we proposed a gradual denoising mechanism, which repeatedly identifies the noisy elements in the current subgraph/subnetwork and replaces them with potential ones in the pruned components within D epochs.

Noisy Component Identification. Given the one-shot masks $\mathbf{M}_g^{\circledcirc}$ and $\mathbf{M}_\theta^{\circledcirc}$, we train the model with fixed sparse masks and a trainable graph mask $\boldsymbol{m}_g$, denoted as $f\left(\{\boldsymbol{m}_g \odot \mathbf{M}_g^{\circledcirc} \odot \mathbf{A}, \mathbf{X}\}, \mathbf{M}_\theta^{\circledcirc} \odot \boldsymbol{\Theta}\right)$ with the objective function similar to Eq. 3. We execute progressive denoising over intervals spanning ΔT epochs, and the d-th epoch

[1] For e_{ij}, we define its edge degree as $(|\mathcal{N}(v_i)| + |\mathcal{N}(v_j)|)/2$.

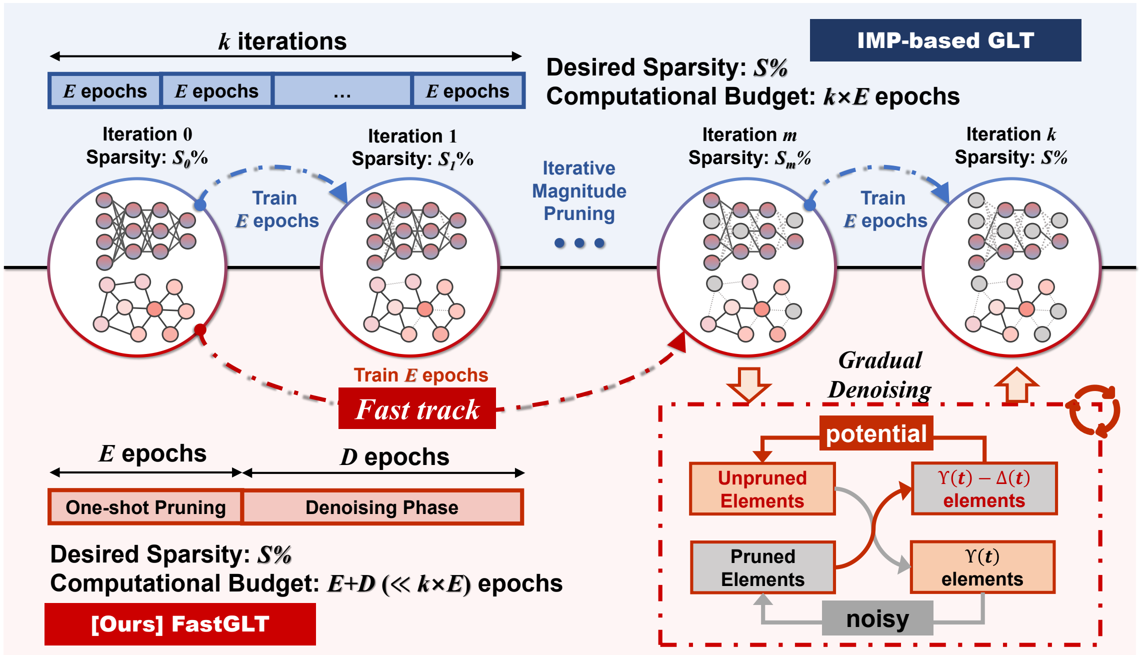

Figure 3: The detailed illustration of `FastGLT` compared to conventional IMP-based GLT. `FastGLT` replaces most of the time-consuming iterative stages with one-shot pruning as a *fast track*, and leverages a gradual denoising module to fine-tune the one-shot tickets to the target sparsity with performance in no way inferior to that of IMP.

is therefore assigned to the $\lceil d/\Delta T \rceil^{th}$ interval. At the end of interval $\mu (1 \leq \mu \leq \lceil D/\Delta T \rceil = \mu^{\text{end}})$, analogous to traditional magnitude pruning, elements with the smallest magnitudes after ΔT training epochs are considered noisy components, as defined as follows:

$$\begin{cases} \mathbf{M}_\theta^{(\text{ns})} = \mathcal{F}\left(-|\mathbf{M}_\theta^{(\mu)} \odot \mathbf{\Theta}_{\Delta T}|,\ \mathbf{N}_\theta^{(\text{ns})}\right) \\ \mathbf{M}_g^{(\text{ns})} = \mathcal{F}\left(-|\mathbf{M}_g^{(\mu)} \odot \boldsymbol{m}_g|,\ \mathbf{N}_g^{(\text{ns})}\right) \end{cases} \tag{5}$$

where $\mathbf{M}_\theta^{(\text{ns})}$ and $\mathbf{M}_g^{(\text{ns})}$ are identified noisy weight/edges, $\mathbf{N}_\theta^{(\text{ns})} = \#\mathbf{M}_\theta^{(\mu)} \times \mathbf{\Upsilon}(\mu)$ and $\mathbf{N}_g^{(\text{ns})} = \#\mathbf{M}_g^{(\mu)} \times \mathbf{\Upsilon}(\mu)$ are the number of identified noisy weight/edges, $\mathcal{F}(m, k)$ returns the indices of top-k elements of matrix m, and $\#$ counts the number of elements in a matrix. $\mathbf{\Upsilon}(\cdot)$ is a denoising scheduler that outputs the ratio of weights/edges to be identified between each interval. Here, we adopt the Inverse Power (Zhu and Gupta 2017; Evci et al. 2020), $\mathbf{\Upsilon}(\mu) = \tau(1 - \mu/\mu^{\text{end}})^{\kappa}$, where τ denotes the initial ratio and κ is the decay factor controlling how fast the ratio decreases with intervals.

Potential Component Discovery. Slightly differently, in unearthing potentially important weights/edges, we adopt new metrics based on our observations in Sec. 2.3. Technically, for pruned weights $\neg\mathbf{M}_\theta^{(\mu)} \odot \mathbf{\Theta}$, we identify those with the highest accumulated gradients as potential ones. For pruned edges $\neg\mathbf{M}_g^{(\mu)} \odot \mathbf{A}$, we regard those with the smallest edge degrees as potential. Notably, to gradually increase the graph/weight sparsity from the initial s_g^{inm} and s_θ^{inm} to the target s_g^{tgt} and s_θ^{tgt}, we ensure that the identified important elements are fewer than the noisy ones by a factor of $\omega_g = \frac{\#||A||_0 \times (s_g^{\text{tgt}} - s_g^{\text{inm}})}{\mu^{\text{end}}}\%$ or $\omega_\theta = \frac{\#\mathbf{\Theta} \times (s_g^{\text{tgt}} - s_g^{\text{inm}})}{\mu^{\text{end}}}\%$ between each interval. The process is defined as follows:

$$\begin{cases} \mathbf{M}_\theta^{(\text{pt})} = \mathcal{F}\left(\sum_{i=1}^{\Delta T} \left|\nabla_{\neg\mathbf{M}_\theta^{(\mu)} \odot \mathbf{\Theta}} \mathcal{L}\right|,\ \mathbf{N}_\theta^{(\text{pt})}\right) \\ \mathbf{M}_g^{(\text{pt})} = \mathcal{F}\left(-(\neg\mathbf{M}_g^{(\mu)} \odot \mathbf{A} \odot \mathbf{S}^{(\mu)}),\ \mathbf{N}_g^{(\text{pt})}\right), \end{cases} \tag{6}$$

where $\mathbf{M}_\theta^{(\text{pt})}$ and $\mathbf{M}_g^{(\text{pt})}$ are potentially important weight/edges discovered from pruned elements, $\mathbf{N}_\theta^{(\text{pt})} = \#\mathbf{M}_\theta^{(\mu)} \times \mathbf{\Upsilon}(t) - \omega_\theta$ and $\mathbf{N}_g^{(\text{pt})} = \#\mathbf{M}_g^{(\mu)} \times \mathbf{\Upsilon}(t) - \omega_g$ are the number of potentially important weight/edges, $\sum_{i=1}^{\Delta T} |\nabla_{\neg\mathbf{M}_\theta^{(\mu)} \odot \mathbf{\Theta}} \mathcal{L}|$ calculates the accumulated gradients of pruned weights in interval μ, and $\mathbf{S}$ is the edge degree matrix, calculated as follows:

$$\mathbf{S} = (\mathbf{D}^{-1}\mathbf{M}_g^{(\mu)}\, \mathrm{d}(\mathbf{D})^{-\frac{1}{2}})(\mathbf{D}^{-1}\mathbf{M}_g^{(\mu)}\, \mathrm{d}(\mathbf{D})^{-\frac{1}{2}})^{\mathrm{T}}, \tag{7}$$

where $\mathbf{D}$ denotes the degree matrix of the sparsed graph $\mathbf{M}_g^{(\mu)}$ and $\mathrm{d}(\mathbf{D})$ returns the node degree vector of $\mathcal{V}$.

Mask Update. Now that we have identified both noisy and potential edges/weights, we proceed to update the graph/weight masks. Specifically, we remove the noisy components from the current mask and incorporate the potential ones, as detailed in the following process:

$$\mathbf{M}^{(\mu+1)} = \left(\mathbf{M}^{(\mu)} \setminus \mathbf{M}^{(\text{ns})}\right) \cup \mathbf{M}^{(\text{pt})}, \tag{8}$$

where $\mathbf{M}^{(\mu+1)}$ is either the updated graph mask $\mathbf{M}_g^{(\mu+1)}$ or weight mask $\mathbf{M}_\theta^{(\mu+1)}$. Note that between each interval, the sparsity of $\mathbf{M}^{(\mu+1)}$ increases by $\left(\frac{s^{\text{tgt}}-s^{\text{inm}}}{\mu_{\text{end}}}\right)\%$ compared to $\mathbf{M}^{(\mu)}$, ensuring that the graph or network precisely reaches the target sparsity at the end of the denoising process.

During the continuous identifying and swapping process, both the network and the graph are denoised to the desired sparsity with satisfactory performance. The overall algorithm framework is showcased in Appendix A.[2]

4 Experiments

In this section, we conduct extensive experiments to answer the following three research questions: (**RQ1**) Can `FastGLT` effectively find graph lottery tickets? (**RQ2**) Can `FastGLT` scale up to larger-scale graphs? (**RQ3**) Does `FastGLT` genuinely accelerate the acquisition of winning tickets and inference speed compared to traditional IMP-based GLT?

4.1 Experiment Setup

Datasets. We select Cora, Citeseer, and PubMed (Kipf and Welling 2016) for node classification. For larger-scale graphs, we opt Ogbn-Arxiv/Proteins/Collab (Hu et al. 2020). For a fair comparison, we follow the datasets splitting criterion used by UGS (Chen et al. 2021b). On small-scale datasets, we use 140 (Cora), 120 (Citeseer), and 60 (PubMed) labeled nodes for training, 500 nodes for validation and 1000 nodes for testing. For OGB datasets, we follow the official splits given in (Hu et al. 2020).

Backbones & Baselines. To assess `FastGLT`'s adaptability across various GNN backbones, we employ three network structures for small-scale datasets: GCN (Kipf and Welling 2017), GIN (Xu et al. 2018) and GAT (Veličković et al. 2017). For larger-scale datasets, we utilize a 28-layer ResGCN (Li et al. 2020a). To comprehensively validate the efficiency of `FastGLT`, we select two state-of-the-art GLT methods, UGS (Chen et al. 2021b) and WD-GLT (Hui et al. 2023), alongside random pruning (RP), for comparison.

Parameter Settings. For small-scale datasets, the hidden dimension is uniformly set to 512. On OGB graphs, we adopt parameter settings similar to UGS (Chen et al. 2021b). Adam is used as the optimizer throughout. To compute intermediate sparsity, we employ $\Psi(s) = s - 0.01s^{1.2}$. The decay factor κ is set as 1 in all experiments. Detailed hyperparameter settings are provided in Appendix B.1.

4.2 Results on Small Graphs ($\mathcal{RQ}1$)

To answer $\mathcal{RQ}1$, we compare our `FastGLT` with UGS, WD-GLT and random pruning on three small-scale datasets for node classification tasks. Following (Wang et al. 2023c,d), toward a clearer illustration, we fix the weight sparsity to zero when investigating how accuracy evolves with the increase of graph sparsity, and vice versa. Fig. 4 illustrates the results on Cora, Citeseer, and PubMed, and we can draw the following observations (**Obs**):

Obs.1. `FastGLT` can find GLTs with sparser subgraph/subnetwork. It is observable that `FastGLT` consistently outperforms other GLT methods across all backbones and benchmarks, attaining improvements in weight sparsity from 1.32% to 45.62% and in graph sparsity from 7.49% to 22.71%. Specifically, on Cora+GIN, the GLT identified by `FastGLT` achieves 28.66% graph sparsity and 89.16% weight sparsity, surpassing WD-GLT by 22.71% and 45.62% respectively.

Obs.2. `FastGLT` demonstrates greater robustness in graph sparsification. `FastGLT` uniquely sustains performance with rising graph sparsity, in contrast to typical GLT methods. On Citeseer+GAT, for instance, while UGS and WD-GLT sharply decline in performance beyond 70% graph sparsity, `FastGLT` remains close to baseline at nearly 90%.

4.3 Results on Large Graphs ($\mathcal{RQ}2$)

To answer $\mathcal{RQ}2$, we conduct comparative experiments on Ogbn-Arxiv, Ogbn-Proteins and Ogbl-Collab with 28-layer ResGCN (Li et al. 2020a). As showcased in Tab. 2 and Tab. 4, we can list the following observations:

Obs. 3. `FastGLT` can scale up to large graphs. `FastGLT` consistently identifies GLTs with weight sparsity over 70% and graph sparsity over 30% across three datasets, surpassing other methods which generally fall below 60% and 30%, respectively. Specifically, `FastGLT` can find GLT with 70.25% weight sparsity or 48.01% graph sparsity on Ogbn-Arxiv, exceeding UGS by 30.18% and 36.82%.

4.4 Efficiency Validation ($\mathcal{RQ}3$)

In this section, we compare the efficiency of `FastGLT` with previous GLT methods from two perspectives: (1) wall-clock time expended in the search for winning tickets, and (2) inference speed of the most sparse tickets discovered. From Tab. 1 and Fig. 5, we draw a conclusion that `FastGLT` *achieves a dual win in GLT search time and computational savings* from the following observations:

Obs.4. `FastGLT` can identify GLTs way faster. Specifically, UGS requires $4.0 - 28.6\times$ the duration of the original dense training to find the sparsest GLT. On GIN, WD-GLT's average time is as high as $105.6\times$. Conversely, `FastGLT` takes only $1.63 - 4\times$ the original dense training time to find the sparsest GLT. Notably, when it comes to finding a ticket with $s_\theta = 49.7\%, s_g = 36.9\%$ on Ogbn-Arxiv (Fig. 5), FastGLT achieves 12.0x and 19.6x acceleration compared to UGS and WD-GLT, respectively.

Obs.5. `FastGLT` excels in obtaining more computationally efficient tickets. Besides faster speed, `FastGLT` also reduces computational load significantly. Across all GCN/GIN/GAT backbones, we achieve over 95% MAC savings, surpassing UGS and WD-GLT by 5.4% to 31.0%.

4.5 Ablation Study & Sensitivity Analysis

Validation of *fast track*. In this part, we validate the premise that *one-shot tickets offer a fast track to winning tickets* from two aspects: convergence speed and optimal performance. Fig. 6 (*Right*) demonstrates that denoising from one-shot tickets not only successfully finds winning tickets, but also offers faster convergence. Tab. 5 compares the maximum s_θ and s_g when starting from randomly initialized tickets versus one-shot tickets. Notably, denoising from random tickets results in up to a 28.94% drop in weight sparsity and a

[2]The algorithm table and other technical appendices are available at https://arxiv.org/abs/2412.07605.

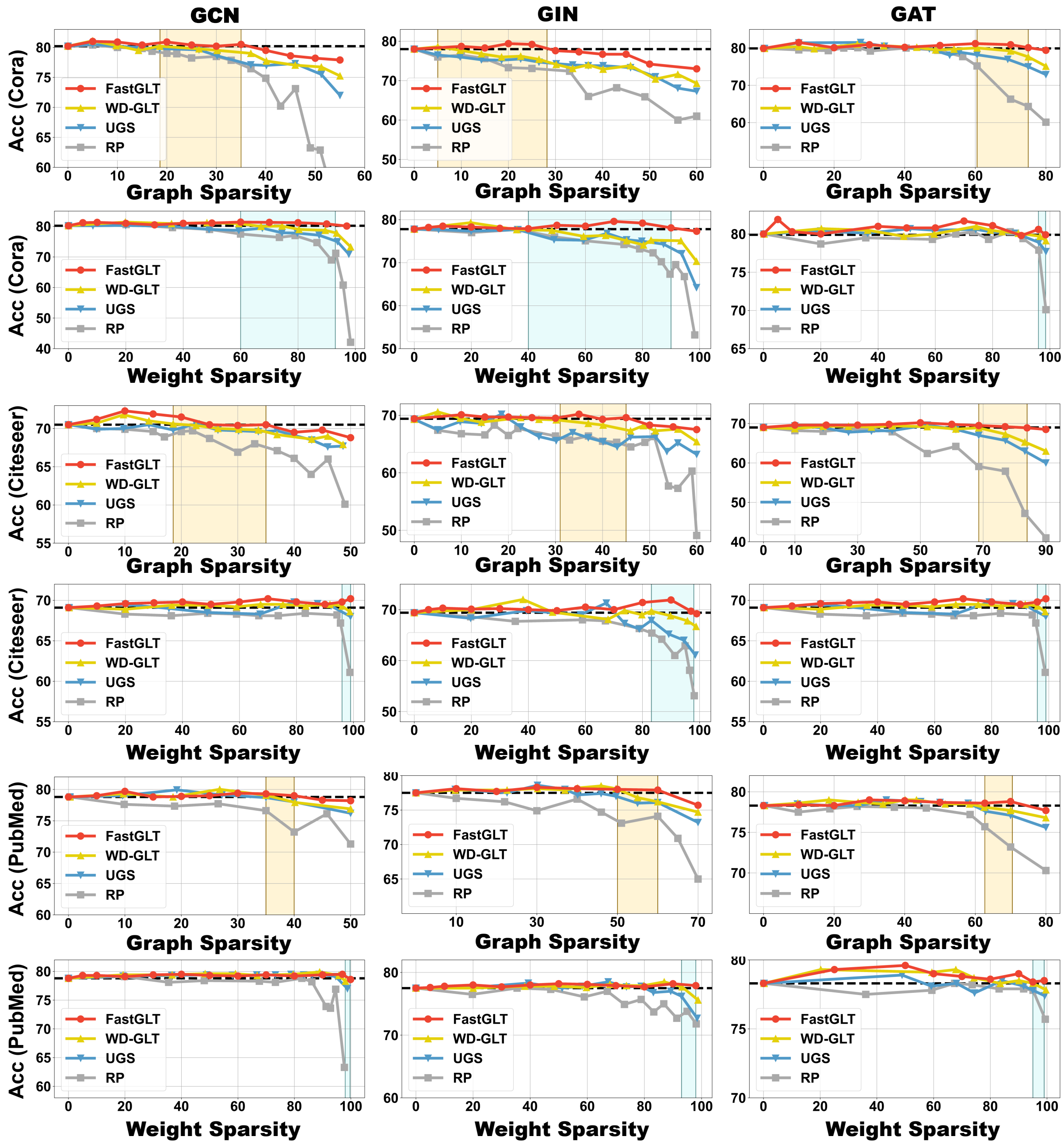

Figure 4: Results of node classification over Cora/Citeseer/PubMed with GCN/GIN/GAT backbones. Black dash lines represent the baseline performance.

24.17% drop in graph sparsity, highlighting the necessity of using one-shot tickets as a fast track.

Effects of denoising interval ΔT. We set graph/weight sparsity at 40%/80%, and explore `FastGLT`'s performances with ΔT values $\{3, 5, 10, 20, 30, 50\}$ on Ogbn-Arxiv+ResGCN and Citeseer+GAT. From Fig. 6 (*Left*), we observe that `FastGLT`'s sensitivity to ΔT is minimal, with a maximum accuracy variance of only 1.35% on Citeseer and 1.74% on Ogbn-Arxiv. The remaining parameter sensitivity analysis is provided in Appendix C.3.

5 Related Work

Lottery Ticket Hypothesis Current LTH methods are categorized into *dense-to-sparse* and *sparse-to-sparse* methods, based on the initial network's over-parameterization. The former starts with a dense network, progressively pruning to the target sparsity (Chen et al. 2021c,a; Ding, Chen, and Wang 2021), while the latter starts with a randomly initialized sparse neural network and dynamically modifies its topology during a single training run through methods like

Model	Methods	Cora			Citeseer			PubMed			Avg.	
		Acc. (%)	Obt. Time	Inf. MACs	Acc. (%)	Obt. Time	Inf. MACs	Acc. (%)	Obt. Time	Inf. MACs	Relative Time	MAC Savings
GCN	Baseline	80.25 ± 0.18	21.4	1996M	70.51 ± 0.06	41.4	6317M	78.80 ± 0.14	1217.3	5077M	1.0×	0.0%
	UGS	80.19 ± 0.06	76.3	817M	70.66 ± 0.08	233.8	1665M	79.01 ± 0.23	3500.9	233M	4.0×	75.6%
	WD-GLT	80.27 ± 0.45	604.3	806M	70.74 ± 0.51	801.7	1541M	78.82 ± 0.33	5634.1	107M	17.4×	80.3%
	`FastGLT`	80.33 ± 0.17	34.9	139M	70.59 ± 0.12	89.7	315M	79.11 ± 0.29	1366.2	50M	1.63×	95.6%
GIN	Baseline	78.06 ± 0.09	7.3	2006M	68.47 ± 0.11	8.6	6328M	77.50 ± 0.17	8.8	5108M	1.0×	0.0%
	UGS	78.17 ± 0.13	39.8	1284M	68.70 ± 0.20	61.0	2073M	77.66 ± 0.30	141.7	438M	28.6×	64.3%
	WD-GLT	78.26 ± 0.44	305.4	1275M	68.50 ± 0.38	891.4	1018M	77.80 ± 0.35	1509.3	331M	105.6×	70.6%
	`FastGLT`	78.32 ± 0.17	20.4	200M	68.54 ± 0.25	21.3	126M	77.59 ± 0.37	17.8	102M	2.4×	95.3%
GAT	Baseline	79.95 ± 0.03	333.1	16059M	69.12 ± 0.18	284.1	50619M	78.35 ± 0.20	920.7	41349M	1.0×	0.0%
	UGS	79.99 ± 0.45	3143.8	1672M	69.23 ± 0.11	3434.1	839M	78.39 ± 0.34	2960.5	3565M	8.2×	93.2%
	WD-GLT	80.11 ± 0.61	5700.7	525M	69.38 ± 1.23	5003.6	568M	78.52 ± 0.20	4533.2	2283M	13.2×	96.4%
	`FastGLT`	80.07 ± 0.37	525.9	204M	69.30 ± 0.21	528.7	414M	78.56 ± 0.68	1270.3	414M	4.8×	98.6%

Table 1: Efficiency comparison among UGS, WD-GLT and `FastGLT`. "Acc. (%)" indicates the accuracy of the sparsest winning tickets obtained; "Obt. Time (s)" represents the wall-clock time consumed to obtain the sparsest winning ticket (for baseline, it refers to the full training time with dense network/graph); "Inference MACs (M)" refers to the inference MACs ($= \frac{1}{2}$FLOPs) required by the identified tickets; "Relative Time (s)" refers to the time relative to the original dense training duration.

Dataset	Ogbn-Arxiv	Ogbn-Proteins	Ogbl-Collab
Graph Sparsity			
Random Pruning	N/A	5.74 ± 2.05	N/A
UGS(Chen et al. 2021b)	11.19 ± 0.42	16.94 ± 0.33	8.20 ± 0.14
WD-GLT(Hui et al. 2023)	30.94 ± 0.51	22.48 ± 0.07	17.14 ± 0.95
`FastGLT` (Ours)	48.01 ± 0.17	34.49 ± 0.13	31.55 ± 0.35

Table 2: Results on 3 large-scale OGB graphs. Each entry denotes the extreme sparsity that a certain method is capable of achieving. *Please note that extreme sparsity refers to the highest sparsity level at which GNN can achieve performance equal to the vanilla GNN.* $\pm$ corresponds to the standard deviation over 5 trials. "N/A" means GLT cannot be found.

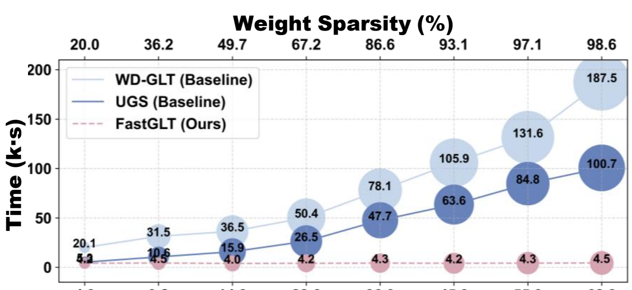

Figure 5: The wall-lock time of UGS, WD-GLT, FastGLT to locate GLTs on Ogbn-Arxiv with different s_g and s_θ.

pruning-and-growing (Mocanu et al. 2018; Wang, Zhang, and Grosse 2020; Liu et al. 2021). It's noteworthy that our gradual denoise mechanism bears a slight resemblance to the topology-adjusting operations of sparse-to-sparse methods. However, our work differs significantly in at least two ways: (1) Current sparse-to-sparse methods focus solely on weight pruning and aren't extendable to joint pruning. In contrast, our denoising mechanism is specifically devised based on observations of weight and graph characteristics; (2) Unlike sparse-to-sparse methods that initiate with a randomly sparse network, we start with one-shot pruning as a fast track to winning tickets, substantiated by extensive ablation studies proving its efficacy.

Graph Lottery Ticket UGS (Chen et al. 2021b) first integrated LTH's concept into GLT research, combining *graph sparsification* with *GNN model compression*. Recent endeavors include GEBT (You et al. 2022), which revealed the existence of graph early-bird tickets. WD-GLT (Hui et al. 2023) enhanced graph pruning through an auxiliary loss function, and DGLT (Wang et al. 2023c) firstly introduced the concept of dual lottery tickets (Bai et al. 2022) to GLT paradigm. However, all these methods fall into the trap of iterative pruning, making the acquisition of GLTs resource-intensive. Conversely, leveraging the one-shot pruning as a fast track, we bypass the extensive computational demands of IMP, acquiring winning tickets much more rapidly.

6 Conclusions

In this work, we propose an effective method termed one-shot pruning and denoising framework toward fast track graph lottery tickets (`FastGLT`), which utilizes one-shot tickets as a *fast track* and denoises them to acquire sparse but performant tickets. Our work reevaluates the relationship between one-shot and IMP tickets, hypothesizing and validating that one-shot tickets can be rapidly denoised to obtain subgraphs/subnetworks that are sparser and perform comparably to IMP-based tickets. This paradigm achieves a significantly faster efficiency in finding GLTs ($1.7 - 44\times$ speedup) compared to previous SOTA methods, offering new insights into how to more rapidly and effectively discover GLTs.

Acknowledgements

This work was supported by the National Key R&D Program of China (Grant no. 2022YFB4501704), the National Science Foundation of China (Grant no. 62472317), the Fundamental Research Funds for the Central Universities, the Shanghai Science and Technology Innovation Action Plan Project (Grant no. 22YS1400600 and 24692118300).

References

Bai, Y.; Wang, H.; Tao, Z.; Li, K.; and Fu, Y. 2022. Dual lottery ticket hypothesis. *arXiv preprint arXiv:2203.04248*.

Chen, J.; Ma, T.; and Xiao, C. 2018. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*.

Chen, T.; Cheng, Y.; Gan, Z.; Liu, J.; and Wang, Z. 2021a. Data-Efficient GAN Training Beyond (Just) Augmentations: A Lottery Ticket Perspective. *Advances in Neural Information Processing Systems*, 34: 20941–20955.

Chen, T.; Sui, Y.; Chen, X.; Zhang, A.; and Wang, Z. 2021b. A unified lottery ticket hypothesis for graph neural networks. In *International Conference on Machine Learning*, 1695–1706. PMLR.

Chen, X.; Zhang, Z.; Sui, Y.; and Chen, T. 2021c. Gans can play lottery tickets too. *arXiv preprint arXiv:2106.00134*.

Chen, Y.; Bian, Y.; Xiao, X.; Rong, Y.; Xu, T.; and Huang, J. 2020. On self-distilling graph neural network. *arXiv preprint arXiv:2011.02255*.

Cheng, D.; Ye, Y.; Xiang, S.; Ma, Z.; Zhang, Y.; and Jiang, C. 2023. Anti-money laundering by group-aware deep graph learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(12): 12444–12457.

Ding, S.; Chen, T.; and Wang, Z. 2021. Audio lottery: Speech recognition made ultra-lightweight, noise-robust, and transferable. In *International Conference on Learning Representations*.

Eden, T.; Jain, S.; Pinar, A.; Ron, D.; and Seshadhri, C. 2018. Provable and practical approximations for the degree distribution using sublinear graph samples. In *Proceedings of the 2018 World Wide Web Conference*, 449–458.

Evci, U.; Gale, T.; Menick, J.; Castro, P. S.; and Elsen, E. 2020. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, 2943–2952. PMLR.

Fang, J.; Zhang, G.; Wang, K.; Du, W.; Duan, Y.; Wu, Y.; Zimmermann, R.; Chu, X.; and Liang, Y. 2024. On regularization for explaining graph neural networks: An information theory perspective. *IEEE Transactions on Knowledge and Data Engineering*.

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

Frankle, J.; Dziugaite, G. K.; Roy, D. M.; and Carbin, M. 2020. Pruning neural networks at initialization: Why are we missing the mark? *arXiv preprint arXiv:2009.08576*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Proceedings of NIPS*.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*.

Hui, B.; Yan, D.; Ma, X.; and Ku, W.-S. 2023. Rethinking Graph Lottery Tickets: Graph Sparsity Matters. In *The Eleventh International Conference on Learning Representations*.

Jin, W.; Zhao, L.; Zhang, S.; Liu, Y.; Tang, J.; and Shah, N. 2021. Graph condensation for graph neural networks. *arXiv preprint arXiv:2110.07580*.

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

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Lee, N.; Ajanthan, T.; and Torr, P. H. 2018. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*.

Li, G.; Xiong, C.; Thabet, A.; and Ghanem, B. 2020a. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*.

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

Liu, S.; Yin, L.; Mocanu, D. C.; and Pechenizkiy, M. 2021. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In *International Conference on Machine Learning*, 6989–7000. PMLR.

Ma, X.; Yuan, G.; Shen, X.; Chen, T.; Chen, X.; Chen, X.; Liu, N.; Qin, M.; Liu, S.; Wang, Z.; et al. 2021. Sanity checks for lottery tickets: Does your winning ticket really win the jackpot? *Advances in Neural Information Processing Systems*, 34: 12749–12760.

Mocanu, D. C.; Mocanu, E.; Stone, P.; Nguyen, P. H.; Gibescu, M.; and Liotta, A. 2018. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1): 2383.

Tailor, S. A.; Fernandez-Marques, J.; and Lane, N. D. 2020. Degree-quant: Quantization-aware training for graph neural networks. *arXiv preprint arXiv:2008.05000*.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *stat*, 1050: 20.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, C.; Zhang, G.; and Grosse, R. 2020. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*.

Wang, K.; Li, G.; Wang, S.; Zhang, G.; Wang, K.; You, Y.; Peng, X.; Liang, Y.; and Wang, Y. 2023a. The Snowflake Hypothesis: Training Deep GNN with One Node One Receptive field. arXiv:2308.10051.

Wang, K.; Liang, Y.; Li, X.; Li, G.; Ghanem, B.; Zimmermann, R.; Yi, H.; Zhang, Y.; Wang, Y.; et al. 2023b. Brave the Wind and the Waves: Discovering Robust and Generalizable Graph Lottery Tickets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wang, K.; Liang, Y.; Wang, P.; Wang, X.; Gu, P.; Fang, J.; and Wang, Y. 2023c. Searching Lottery Tickets in Graph Neural Networks: A Dual Perspective. In *The Eleventh International Conference on Learning Representations*.

Wang, K.; Zhang, G.; Zhang, X.; Fang, J.; Wu, X.; Li, G.; Pan, S.; Huang, W.; and Liang, Y. 2024. The heterophilic snowflake hypothesis: Training and empowering gnns for heterophilic graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3164–3175.

Wang, L.; Huang, W.; Zhang, M.; Pan, S.; Chang, X.; and Su, S. W. 2022. Pruning graph neural networks by evaluating edge properties. *Knowledge-Based Systems*, 256: 109847.

Wang, Y.; Liu, S.; Chen, K.; Zhu, T.; Qiao, J.; Shi, M.; Wan, Y.; and Song, M. 2023d. Adversarial Erasing with Pruned Elements: Towards Better Graph Lottery Ticket. *arXiv preprint arXiv:2308.02916*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31.

You, H.; Lu, Z.; Zhou, Z.; Fu, Y.; and Lin, Y. 2022. Early-bird gcns: Graph-network co-optimization towards more efficient gcn training and inference via drawing early-bird lottery tickets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 8, 8910–8918.

Zhang, G.; Sun, X.; Yue, Y.; Jiang, C.; Wang, K.; Chen, T.; and Pan, S. 2024a. Graph Sparsification via Mixture of Graphs. *arXiv preprint arXiv:2405.14260*.

Zhang, G.; Wang, K.; Huang, W.; Yue, Y.; Wang, Y.; Zimmermann, R.; Zhou, A.; Cheng, D.; Zeng, J.; and Liang, Y. 2024b. Graph lottery ticket automated. In *The Twelfth International Conference on Learning Representations*.

Zhang, G.; Yue, Y.; Wang, K.; Fang, J.; Sui, Y.; Wang, K.; Liang, Y.; Cheng, D.; Pan, S.; and Chen, T. 2024c. Two heads are better than one: Boosting graph sparse training via semantic and topological awareness. *arXiv preprint arXiv:2402.01242*.

Zhang, M.; and Chen, Y. 2018. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31.

Zhang, M.; and Chen, Y. 2019. Inductive matrix completion based on graph neural networks. *arXiv preprint arXiv:1904.12058*.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, 1.

Zhang, Z.; Chen, X.; Chen, T.; and Wang, Z. 2021. Efficient lottery ticket finding: Less data is more. In *International Conference on Machine Learning*, 12380–12390. PMLR.

Zhou, H.; Srivastava, A.; Zeng, H.; Kannan, R.; and Prasanna, V. 2021. Accelerating large scale real-time GNN inference using channel pruning. *arXiv preprint arXiv:2105.04528*.

Zhu, M.; and Gupta, S. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*.