Learning Backbones: Sparsifying Graphs through Zero Forcing for Effective Graph-Based Learning

Obaid Ullah Ahmad[1], Anwar Said[2] Mudassir Shabbir[3], Xenofon Koutsoukos[2], and Waseem Abbas[1]

[1] University of Texas at Dallas, Richardson, TX, USA
ObaidUllah.Ahmad@utdallas.edu,
[2] Vanderbilt University, Nashville, TN, USA
[3] Information Technology University, Lahore, Pakistan

Abstract. This paper introduces a novel framework for graph sparsification that preserves the essential learning attributes of original graphs, improving computational efficiency and reducing complexity in learning algorithms. We refer to these sparse graphs as "learning backbones." Our approach leverages the zero-forcing (ZF) phenomenon, a dynamic process on graphs with applications in network control. The key idea is to generate a tree from the original graph that retains critical dynamical properties. By correlating these properties with learning attributes, we construct effective learning backbones. We evaluate the performance of our ZF-based backbones in graph classification tasks across eight datasets and six baseline models. The results demonstrate that our method outperforms existing techniques. Additionally, we explore extensions using node distance metrics to further enhance the framework's utility.

Keywords: Sparsification, Network Control Backbone, Graph Neural Networks, Graph Classification

1 Introduction

In recent decades, networks have become essential for analyzing complex systems with applications in computer vision [18], 3D object modeling [11], and chemical molecules [16]. In machine learning, constructing efficient graph representations is critical for tasks like social network analysis, financial systems, and recommendation systems [25]. The complexity of real-world graphs often requires extracting sparse yet informative substructures, known as graph learning backbones, to enable effective learning [26]. This paper addresses the challenge of identifying these sparse representations while retaining essential properties for downstream tasks by integrating principles from Network Control Theory [10].

Control theory, renowned for analyzing and steering dynamic systems [22], helps select minimal edge sets that capture a graph's intrinsic behavior [5, 17]. Viewing a graph as a dynamic system, this approach ensures controllability by maintaining key structural properties.

In graph-based learning, preserving dynamic properties is crucial for accurate classification and prediction [20]. Previous methods such as graph sparsifiers and spanners [13, 14] aimed to reduce graph complexity while retaining key properties, but often lacked alignment with specific graph learning objectives [26]. Techniques like spectral rewiring [6], Forman curvature-based rewiring [23], and graph diffusion [8] have also been explored for optimizing graph structures. However, these methods can sometimes lead to information loss or significant graph densification, which complicates learning tasks. Sparse subgraph extraction techniques have focused on community preservation [26] but remain limited in balancing both sparsity and learning effectiveness.

Inspired by tree-like substructures in communication networks [29], we propose that a connected tree subgraph represents the minimal structure required for learning. These sparse trees, derived from control theory, offer efficient representations while preserving critical properties.

Graph learning has advanced significantly [20], but determining the ideal graph structure for specific learning objectives remains a challenge [24]. This motivates exploration of the Graph Lottery Ticket Hypothesis (GLTH), which posits that within any complex graph, there exists a sparse substructure capable of achieving comparable performance to the full graph [24], opening new avenues for scalable graph learning [7].

However, prevailing methods for uncovering these winning tickets often rely on pruning or sampling, risking information loss [7]. This paper introduces a novel control-theoretic approach to discovering these tickets. The Graph Lottery Ticket Hypothesis is articulated as follows:

Graph Lottery Ticket Hypothesis [24]: For any given graph, there exists a sparse subset of edges such that training any graph learning algorithm solely on this subset yields performance comparable to that of the original graph.

We present a detailed exploration of our control-theoretic approach to discovering Graph Lottery Tickets (GLTs), which we refer to as learning backbones. We propose that the zero-forcing set (ZFS)-based control backbone [2], a tree, represents the winning ticket. Our method demonstrates superior precision and efficiency in identifying substructures compared to existing techniques. Additionally, we extend this concept by preserving other control properties in the graph. Through experiments on diverse datasets and tasks, we showcase the exceptional performance and sparsity of the winning tickets identified by our approach.

The rest of the paper is organized as follows: Section 2 introduces important notations and formulates the main problem of graph sparsification for graph classification. Section 3 defines the concept of the ZFS-based backbone and proposes several approaches to compute the learning backbone using control properties of networks. Section 4 presents empirical results for graph classification. Finally, Section 5 concludes the paper and discusses future directions.

In the next section, we review some notations to be used in the rest of the paper and explain the main problem.

2 Preliminaries and Problem Formulation

In this section, we establish the fundamental notation to be employed throughout the paper and properly formulate the main problem addressed in this paper.

2.1 Preliminaries

An undirected graph $G = (V, E)$ represents a multi-agent network, where the vertex set V represents agents and the edge set $E \subseteq V \times V$ denotes interactions between them. An edge between vertices u and v is denoted by the unordered pair (u, v). The *neighborhood* of vertex u is defined as $\mathcal{N}_G(u) = \{v \in V : (u, v) \in E\}$, and the *degree* of u is $\deg(u) = |\mathcal{N}_G(u)|$. The *average degree* $\bar{d}$ is given by $\bar{d} = \frac{1}{|V|} \sum_{v \in V} \deg(v)$, where $\deg(v)$ is the degree of vertex v.

A *path* P in G is a sequence of distinct vertices $(v_1, v_2, \ldots, v_k)$ such that for each i from 1 to $k-1$, there exists an edge between v_i and v_{i+1}. The *distance* between vertices u and v, denoted $d_G(u, v)$, is the number of edges in the shortest path between u and v. For simplicity, the subscript is dropped when the context is clear. A graph $\hat{G} = (\hat{V}, \hat{E})$ is a *subgraph* of $G = (V, E)$, denoted $\hat{G} \subseteq G$, if $\hat{V} \subseteq V$ and $\hat{E} \subseteq E$.

A *connected component* C of G is a maximal subset of vertices $V' \subseteq V$ such that for every pair of vertices u, v in V', there exists a path between u and v. A *tree* is an undirected graph that is connected and acyclic, or equivalently, a graph with n vertices and $n-1$ edges.

2.2 Problem Formulation

In the context of graph-based machine learning, the graph classification problem is a fundamental task. The objective is to assign a discrete label to an entire graph, indicating the class to which the graph belongs. This task finds applications in various domains, such as cheminformatics, where graphs represent molecules, and social network analysis, where graphs represent interactions between individuals [12].

Formally, given a collection of graphs $\{G_1, G_2, \dots, G_k\}$, where each graph $G_i = (V_i, E_i)$ consists of a set of vertices V_i and a set of edges E_i, and a corresponding set of labels $\{y_1, y_2, \dots, y_k\}$ with $y_i \in \{0, 1, \dots, C\}$, the goal is to learn a function $\phi : \mathcal{G} \to y$, where $\mathcal{G}$ is the set of all possible graphs, $y \in \{0, 1, \dots, C\}$, and $C \in \mathbb{Z}$ is the number of possible labels. The function ϕ takes an input graph G_i and outputs a label $\tilde{y}_i$, representing the predicted class of the graph. A machine learning approach to this problem involves training a model to generate this discrete labeling: a model $\phi(G_i, \boldsymbol{\theta})$ that takes an input graph G_i and outputs a probability score $\phi : \{G_i\} \to [0,1]^C$ indicating the likelihood of each graph being classified as one of the classes, where $\boldsymbol{\theta}$ are the learnable weights. The learned function $\phi(G, \boldsymbol{\theta})$ should minimize the classification error $\mathcal{L}(y, \tilde{y})$ on a given set of graphs, where the error is essentially the difference between the predicted label $\tilde{y}$ and the true given label y.

Problem 1 ***(Graph Classification):*** *Given a graph $G = (V, E)$, the goal is to map G to a discrete label $y \in \{0, 1, \dots, C\}$ using a machine learning model with learnable weights $\boldsymbol{\theta}$. The mapping function ϕ can be expressed as:*

$$\tilde{y} = \phi(G; \boldsymbol{\theta}),$$

where $\tilde{y}$ is the predicted label for the graph G and $\boldsymbol{\theta}$ represents the parameters of the model.

In many applications of graph-based machine learning, dealing with large and dense graphs can pose significant computational challenges. Graph sparsification is a crucial technique to address this issue, aiming to reduce the number of edges in a graph while preserving its essential properties [24]. By simplifying the graph structure, sparsification can lead to more efficient algorithms, reduced memory usage, and faster processing times, without significantly compromising the performance of graph-based tasks such as classification. Formally, given a graph $G = (V, E)$, a sparsification function $\mathcal{A}$ produces a sparser subgraph $\hat{G} = (V, \hat{E})$ such that $\hat{E} \subseteq E$ and $|\hat{E}| \ll |E|$. The goal is to ensure that $\hat{G}$ retains the key structural properties of G necessary for downstream machine learning tasks.

Problem 2 ***(Graph Sparsification):*** *Given a graph $G = (V, E)$ and a label y, the goal is to find a sparsification function $\mathcal{A} : \mathcal{G} \to \hat{\mathcal{G}}$ such that $G \mapsto \hat{G} = (V, \hat{E})$, $\hat{E} \subseteq E$, and $\phi(\hat{G}, \boldsymbol{\theta}) = \tilde{y}$ where the predicted label $\tilde{y}$ should be the same as the given label y.*

In light of graph classification, the problem can be framed to incorporate graph sparsification. To leverage sparsification, we first apply a sparsification function $\mathcal{A}$ to each graph G_i, obtaining a sparser graph $\hat{G}_i = \mathcal{A}(G_i) \quad \forall i \in \{1, 2, \dots, k\}$. Then, we learn the classification function ϕ on the set of sparser graphs $\{\hat{G}_1, \hat{G}_2, \dots, \hat{G}_k\}$. The objective is to minimize the classification error $\mathcal{L}(y, \tilde{y})$ on the sparsified graphs, ensuring $\phi(\hat{G}, \boldsymbol{\theta}) = \tilde{y} \approx y$ and thus maintaining high classification performance on the original graph set. This idea is presented in Figure 1 where the gray box represents the main focus of this work.

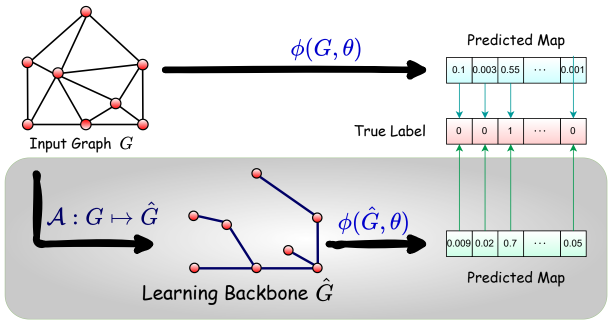


Fig. 1: Main Idea: Sparsify the graph while maintaining the critical learning backbone for downstream machine learning tasks such as graph classification. The predicted label $\phi(\hat{G}, \boldsymbol{\theta}) = \tilde{y}$ should be close to the true label y.

In Section 3.2, we propose a novel approach to sparsify a graph for computing the learning backbone. The sparse graph we propose is a tree, as it is the minimum connected graph structure. Finding a tree graph for a given graph is not a trivial task, as it can be computationally expensive to identify a suitable tree that preserves the essential properties of the original graph required for downstream learning tasks. The exact number of spanning trees of a given graph can be computed by the Matrix Tree Theorem [9]. The number of spanning trees of a given graph $G = (V, E)$ is the normalized product of the non-zero eigenvalues of the Laplacian matrix, and it can be as large as $\left(\frac{2m-\Delta-\delta-1}{n-3}\right)^{n-3}$, where n is the number of nodes and $n > 3$, m is the number of edges, Δ is the maximum degree, and δ is the minimum degree [15]. Note that the number of spanning trees in a given graph can be exponential with respect to the number of vertices.

In the next section, we present our proposed approach for finding a learning tree backbone.

3 Learning Backbone

Control theory is a branch of engineering and mathematics focused on the behavior of dynamical systems with inputs. Its goal is to develop a control strategy that governs the system's output by manipulating the inputs. A core concept in control theory is feedback, where the system's output is measured and used to adjust the inputs to maintain desired performance. Control theory has applications in robotics, aerospace, and economics. The structural properties of the underlying graph, representing the network of interconnected components, significantly influence a system's dynamic behavior. For instance, the presence or absence of specific edges can affect the stability, controllability, and observability of the system [1].

The dynamic behavior of a control system is closely related to graph-based machine learning. In machine learning, how information propagates through a graph is crucial for tasks such as node classification, link prediction, and graph classification [20, 21]. The graph structure dictates how signals spread across the network, influencing the performance of graph neural networks (GNNs) and other models. Sparse representations, like trees, play a vital role by preserving essential control properties, such as connectivity and controllability [2], while reducing computational complexity. This approach aligns with the Graph Lottery Ticket Hypothesis, which suggests that a sparse substructure within a complex graph can achieve comparable performance to the original, optimizing both control and learning objectives.

In this section, we explore the concept of the network controllability backbone, which aims to identify a sparse subset of edges that preserves the network's controllability under structural perturbations. We begin by establishing the fundamental framework for understanding controllability in networked systems and then introduce the concept of a learning backbone.

3.1 Controllability Framework

Consider a network of n agents, denoted by $V = \{v_1, v_2, \cdots, v_n\}$. Among these agents, m are designated as input or leader vertices, represented as $V_\ell = \{\ell_1, \ell_2, \cdots, \ell_m\} \subseteq V$, while the remaining vertices act as followers. The network's dynamics are modeled by the following linear time-invariant system:

$$\dot{x}(t) = Mx(t) + Hu(t), \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the state vector, and $u(t) \in \mathbb{R}^m$ represents the external input injected through the m leaders. The matrix $M \in \mathcal{M}(G)$ is the system matrix associated with the graph G, and $H \in \mathbb{R}^{n\times m}$ is the input matrix determined by the leader vertices. The family of matrices $\mathcal{M}(G)$ is defined as follows:

$$\begin{aligned}\mathcal{M}(G) = \{M \in \mathbb{R}^{n\times n} \ : \ M = M^\top, \text{ and for } i \neq j,\\ M_{ij} \neq 0 \Leftrightarrow (i,j) \in E(G)\}.\end{aligned} \tag{2}$$

This definition encompasses a broad class of system matrices associated with the graph G, including the adjacency matrix, Laplacian matrix, and the signless Laplacian matrix.

A system (1) is *controllable* if an input $u(t)$ can drive the system from any initial state $x(t_0)$ to any desired state $x(t_f)$ in finite time. We say that (M, H) is a *controllable pair* if and only if the controllability matrix $\mathcal{C}(M,H) \in \mathbb{R}^{n\times nm}$ is full rank, i.e., $\texttt{rank}(\mathcal{C}(M,H)) = n$. The controllability matrix is given by:

$$\mathcal{C}(M,H) = \begin{bmatrix} H & MH & M^2H & \cdots & M^{n-1}H \end{bmatrix}. \tag{3}$$

Definition 1. *(Strong Structural Controllability (SSC)) A graph $G = (V,E)$ with a specified set of leaders $V_\ell \subseteq V$ (and the corresponding H matrix) is* strongly structurally controllable *if and only if (M,H) is a controllable pair for all $M \in \mathcal{M}(G)$.*

If the network G is strongly structurally controllable for a given set of leaders, then the rank of the controllability matrix does not depend on the edge weights (as long as they satisfy the conditions given by $\mathcal{M}(G)$). For the remainder of this paper, we will refer to strong structural controllability simply as *controllability.* The dimension of the strongly structurally controllable subspace, denoted by $\gamma(G, V_\ell)$, is the smallest possible rank of the controllability matrix under feasible edge weights.

Network Controllability Backbone The main idea of a controllability backbone is to identify a minimal subset of edges within a network that ensures the preservation of its controllability in any subgraph. We define the *controllability backbone* as this sparse subgraph, denoted by B, such that any subgraph $\hat{G}$ containing B maintains at least the same level of controllability as the original network G.

Definition 2. *(Controllability Backbone) For a given graph $G = (V,E)$ and a set of leaders $V_\ell \subseteq V$, the controllability backbone $B = (V, E_B)$ is a subgraph of G such that any subgraph $\hat{G} = (V, \hat{E})$ containing E_B, i.e., $E_B \subseteq \hat{E} \subseteq E$, satisfies:*

$$\gamma(\hat{G}, V_\ell) \geq \gamma(G, V_\ell). \tag{4}$$

In essence, the controllability backbone ensures that the controllability of any subgraph encompassing it does not deteriorate compared to the original network.

3.2 Zero Forcing for Controllability Backbone

Zero forcing is a rule-based coloring technique for vertices in a graph, providing a lower bound on the dimension of Strong Structural Controllability (SSC). By leveraging zero forcing, our aim is to identify a subset of edges constituting the controllability backbone, termed as the ZFS-based backbone.

Definition 3 (***Zero Forcing (ZF) Process***)**.** *Let $G = (V, E)$ be a graph where each vertex $v \in V$ is initially colored either* `black` *or* `white`*. The ZF process iteratively changes the color of* `white` *vertices to* `black` *according to the following rule until no further color changes are possible:* Color change rule: If a `black` vertex $v \in V$ has exactly one `white` neighbor u, change the color of u to `black`.

We define a *forced* relationship between vertices v and u if a `black` vertex v changes the color of a `white` vertex u to `black` during the ZF process.

Definition 4 (***Derived Set***)**.** *Let $G = (V, E)$ be a graph with $V_\ell \subseteq V$ representing the initial set of* `black` *vertices. The derived set [4], denoted by $dset(G, V_\ell)$, is the set of* `black` *vertices obtained after the ZF process, and $|dset(G, V_\ell)| = \zeta(G, V_\ell)$. When the context is clear, we omit the parameter V_ℓ.*

The set of initial `black` vertices V_ℓ is also known as the *input or leader set*. For a given V_ℓ, $dset(G, V_\ell)$ is unique [4]. Now, we define the zero forcing set.

Definition 5 (***Zero Forcing Set (ZFS)***)**.** *For a graph $G = (V, E)$, $V_\ell \subseteq V$ is a ZFS if and only if $dset(G, V_\ell) = V$. We denote a ZFS of G by $Z(G)$.*

Figure 2 illustrates zero forcing through a set of input vertices and the corresponding derived set. Initially, $V_\ell = \{v_1, v_2, v_5, v_6\}$ are colored black. In the next step, v_2 can force v_3 as it is its only white neighbor and so on.

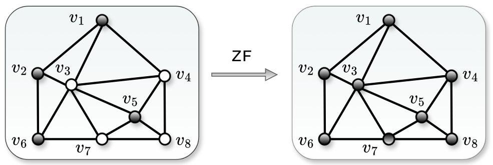


Fig. 2: $V_\ell = \{v_1, v_2, v_5, v_6\}$ is the input set. After the ZF process, $dset(G, V_\ell) = V$, as indicated by the `black` vertices. Hence, V_ℓ is a ZFS.

The zero forcing phenomenon is significant in characterizing the network's SSC [28]. In particular, the size of the derived set for a given set of input vertices provides a lower bound on the dimension of SSC, i.e., for a network $G = (V, E)$ with the leader set $V_\ell \subseteq V$, we have $\zeta(G, V_\ell) \leq \gamma(G, V_\ell)$ [3,28]. By computing the ZFS, we obtain a lower bound on the dimension of SSC, facilitating the identification of the controllability backbone.

ZFS-based Backbone Our goal is to discover a backbone that maintains the zero forcing bound $\zeta(G, V_\ell)$ for a given leader set V_ℓ. The objective is to identify a subset of edges E_{B_Z} in the graph $G = (V, E)$ with V_ℓ such that the ZFS-based controllability bound is preserved in any subgraph $\hat{G} = (V, \hat{E})$ containing those edges ($E_{B_Z} \subseteq \hat{E}$). Formally, we define the ZFS-based backbone as follows:

Definition 6. *(*ZFS-based Backbone*) Given a graph $G = (V, E)$ and a leader set V_ℓ, the ZFS-based backbone, denoted as $B_z = (V, E_{B_z})$, is a subgraph where any subgraph $\hat{G} = (V, \hat{E})$ containing E_{B_Z} satisfies $\zeta(\hat{G}, V_\ell) \geq \zeta(G, V_\ell)$.*

Thus, in any subgraph of G containing the ZFS-based backbone, the dimension of SSC is at least $\zeta(G, V_\ell)$, i.e., $\gamma(\hat{G}, V_\ell) \geq \zeta(G, V_\ell)$.

3.3 Controllability Backbone as Learning Backbone

In graph-based learning, optimizing the underlying graph structure for both control and information propagation is crucial. The Zero Forcing Set (ZFS) method is a powerful tool for maintaining network controllability. By leveraging the principles of strong structural controllability (SSC), we can identify a minimal subset of edges, termed the ZFS-based controllability backbone, which preserves the essential control properties of the original graph. This backbone, effectively forming a sparse substructure, ensures robust dynamic behavior while significantly reducing computational complexity. Using the ZFS-based controllability backbone as the learning backbone aims to enhance the efficiency of graph classification tasks while maintaining the critical control properties of the original network.

It has been shown that the ZFS-based backbone B_z is a set of paths originating from vertices $v \in V_\ell$, always having $n - |V_\ell|$ edges, and consequently, $|V_\ell|$ connected components [2]. If the original graph is connected, edges can be added to form a subgraph $G' = (V, E')$ such that $E_{B_Z} \subseteq E' \subseteq E$, and G' is a connected tree. This tree, known as the learning backbone, substitutes the original tree and can be used for downstream machine-learning tasks. This approach is presented in Algorithm 1.

Algorithm 1 Computing Learning Backbone

```
Input: Graph $G = (V, E)$
Output: Learning backbone $\hat{G} = (V, \hat{E})$
1: Compute a zero-forcing set $V_\ell$ [3]
2: Initialize a graph $B_z$ with paths originating from all vertices $v \in V_\ell$ by running the zero-forcing
   process
3: Add any $|V_\ell| - 1$ edges from $G$ to $B_z$ to form $\hat{G}$ such that $\hat{G}$ becomes a connected tree
```

Theorem 1. *Given a graph $G = (V, E)$, Algorithm 1 returns a learning backbone, a* connected tree*, that is strong structurally controllable for the computed leader set V_ℓ.*

Proof. For any given graph $G = (V, E)$ and leader set V_ℓ, any subgraph $\hat{G} = (V, \hat{E})$, where $E_{B_Z} \subseteq \hat{E} \subseteq E$, satisfies the relation

$$\zeta(\hat{G}, V_\ell) \geq \zeta(G, V_\ell)$$

by definition. In step 1 of Algorithm 1, we compute a zero-forcing set that makes the graph fully controllable. Hence, $\gamma(G, V_\ell) = |V|$. The ZFS-based backbone B_z contains an unconnected set of paths where each path originates from a leader vertex. B_z can be computed from Algorithm 1 of [2]. By definition of B_z, we can add any number of edges from the original graph randomly, and the graph will remain fully controllable. The learning backbone $\hat{G}$ contains only the edges that are in the original graph besides containing the controllability backbone B_z. Hence, we can compute a connected tree with $n - 1$ edges from Algorithm 1 where the tree would be strong structurally controllable for the computed zero-forcing set V_ℓ.

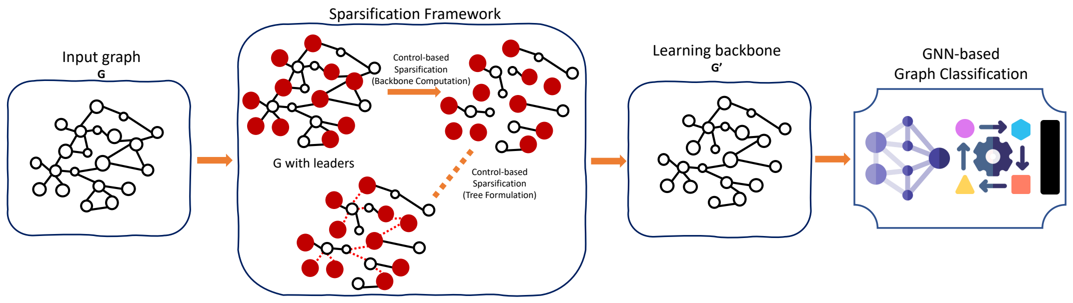


Fig. 3: Illustration of the proposed framework: The process begins by identifying a leader vertex subset within the graph. Using network control theory, a graph sparsification framework is then applied to derive a tree-like structure, called the ‘learning backbone’, from the original graph.

3.4 Generalized Learning Backbone

In networked systems, preserving various control properties is crucial for robust performance and effective information propagation. While the Zero Forcing Set (ZFS) method ensures network controllability, other control properties, such as controllability matrices, graph distances, and structural patterns, also significantly impact dynamical system behavior.

Graph distances, representing the shortest paths between vertices, are particularly important for understanding how control signals propagate through a network. The distribution of these distances influences system stability and responsiveness [28]. In closely connected networks, control inputs more efficiently affect the entire system. The distances between all vertices and leaders also determine the lower bound on controllability rank, $\gamma(G, V_\ell)$ [27]. For a network $G = (V, E)$ with leaders $V_\ell = \{\ell_1, \ell_2, \cdots, \ell_m\}$, the *distance-to-leader (DL) vector* for each $v_i \in V$ is defined as

$$D_i = \left[\, d(\ell_1, v_i)\; d(\ell_2, v_i) \cdots d(\ell_m, v_i) \,\right]^T \in \mathbb{Z}^m,$$

where $[D_i]_j = d(\ell_j, v_i)$ is the distance between leader ℓ_j and vertex v_i. The maximum sequence of these DL vectors that meets certain constraints defines the lower bound on $\gamma(G, V_\ell)$.

Ahmad et al. introduced the distance-based controllability backbone $B_d = (V, E_{B_d})$, which emphasizes preserving key distances between vertex pairs, unlike the ZFS-based backbone B_z, which focuses on tree-like structures [2]. While B_z ensures controllability via paths, B_d maintains critical distances while preserving sparsity with $O(n)$ edges, where n is the number of vertices.

Incorporating graph distances into backbone construction enhances control properties and ensures robust dynamic behavior, supporting downstream tasks like graph classification by retaining the network’s structural integrity, as demonstrated in Section 4.

In summary, while the ZFS-based method is valuable for ensuring controllability, considering additional control properties like graph distances offers a more comprehensive approach. The distance-based backbone balances sparsity with the preservation of critical features, providing robust control across various applications. We incorporate the distance-based backbone in our experiments, detailed in Section 4.

4 Experimental Results

In this section, we offer a comprehensive evaluation of the proposed framework within the context of graph classification tasks, employing real-world social networks and molecular

Table 1: Dataset stats

Dataset	# of Graphs	# of Nodes		Average Degree $\bar{d}$		Density			
						Original		Backbone	
		min	max	Original	Backbone	min	max	min	max
MUTAG	188	10	28	2.189	1.88	0.082	0.222	0.071	0.20
PTC	344	2	64	1.981	1.862	0.034	1.0	0.031	1.0
PROTEINS	1113	4	620	3.735	1.893	0.005	1.0	0.003	0.5
NCI1	4110	3	111	2.155	1.908	0.019	0.667	0.018	0.667
Deezer Ego	9,629	11	363	4.292	1.887	0.015	0.909	0.006	0.182
GitHub Stargazers	12,725	10	957	3.111	1.939	0.003	0.561	0.004	0.200
Twitch Ego	127,094	12	52	5.397	1.922	0.038	0.967	0.038	0.143
Reddit Threads	203,088	11	97	2.039	1.889	0.021	0.328	0.021	0.182

datasets. We introduce the datasets and then provide a detailed description of the experimental setup. Following the setup, we discuss the results, elucidating the efficacy of our framework.

4.1 Datasets

We evaluate our proposed approach using eight real-world datasets relevant for binary graph classification tasks. These datasets include MUTAG, PTC, PROTEINS, NCI1, Deezer Ego Network, GitHub Stargazers, Twitch Ego Networks, and Reddit Threads [16, 19]. Each dataset presents unique challenges in graph classification, offering a comprehensive testbed for assessing the effectiveness of our ZFS-based backbone approach.

4.2 Experimental Setup

We evaluate six widely recognized graph convolution methods: k-GNN, GraphSAGE, GCN, Transformer Convolution (UniMP), Residual Gated Graph ConvNets (ResGatedGCN), and Graph Attention Network (GAT). The proposed learning frameworks consists of three GNN layers, each with 64 hidden units. After the GNN layers, we apply Sort Aggregation, followed by two 1D convolution layers with Max Pooling. The output is then passed through a two-layer multi-layer perceptron, each layer containing 32 hidden neurons.

For evaluation, we perform 10-fold cross-validation, training each model for 100 epochs. The learning rate is set to 1×10^{-4}, and weight decay is 5×10^{-4}. All experiments are conducted on a Lambda machine with an AMD Ryzen Threadripper PRO 3975WX 32-Core CPU, 512 GB of RAM, and an NVIDIA RTX 3090 GPU with 16 GB of memory.

We present the ROC AUC (Receiver Operating Characteristic Area Under the Curve) classification results for all eight datasets, comparing the original graphs and the ZFS-based tree backbone graphs in Table 2. Consistent architectures and experimental settings are used for evaluation. Overall, the performance between the backbone graphs B_z and the original graphs is comparable across all datasets.

4.3 Results Analysis

For certain datasets, such as Deezer Ego and PTC, we observe from Table 2 that the ZFS-based backbone—derived from zero forcing-based controllability—serves as a more effective representation for graph learning tasks. This is exemplified by the notable performance improvement seen with the PTC dataset, where the application of a UniMP baseline GNN model on the backbone graph resulted in a maximum performance enhancement of 9.04%.

Table 2: Comparison of ROC AUC scores of the proposed method (ZFS-based backbone) against original graphs. Pairs where the backbone ROC AUC is within 5% of the original are highlighted in blue. Additionally, backbone values higher than the original are **bolded**.

Datasets	**k-GNN**		**SAGE**		**GCN**		**UniMP**		**ResGatedGCN**		**GAT**	
	Original	Backbone	Original	Backbone	Original	Backbone	Original	Backbone	Original	Backbone	Original	Backbone
Deezer Ego	50.98	**52.07**	50.82	**52.54**	48.45	**52.66**	50.34	**52.65**	51.72	**54.21**	50.54	**50.88**
Twitch Ego	72.23	**72.36**	72.34	**72.47**	72.44	72.25	72.35	**72.47**	72.42	**72.49**	72.37	**72.47**
GitHub Stargazers	71.47	68.54	64.95	61.85	65.58	64.65	65.59	65.45	72.55	68.52	65.01	62.03
Reddit Threads	83.80	83.40	82.99	**83.45**	83.06	**83.26**	83.87	83.54	83.87	83.55	83.84	83.05
MUTAG	93.20	90.13	86.02	**92.95**	88.07	**92.31**	91.92	82.94	92.95	92.95	90.25	**95.13**
PTC	49.10	**56.80**	47.79	**56.07**	50.93	46.87	48.36	**57.40**	57.53	48.70	53.33	**57.07**
PROTEINS	78.65	75.62	77.59	73.78	78.37	72.36	77.86	76.25	77.02	75.34	77.62	72.95
NCI1	77.78	69.60	69.23	67.92	72.34	61.66	70.63	67.63	72.50	66.41	71.72	66.89

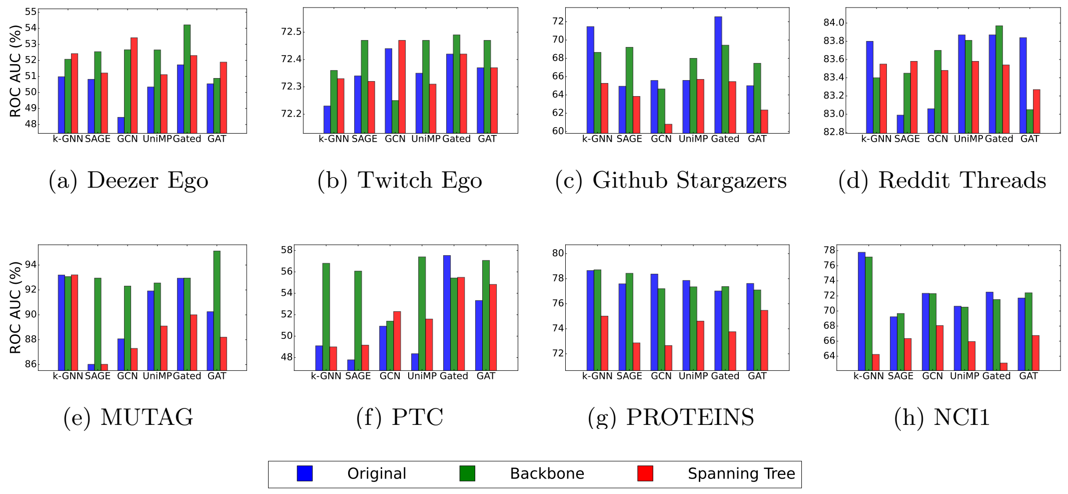
In fact, *in 20 out of 48 combinations, the backbone representation resulted in a better ROC AUC compared to the original graphs.* Moreover, *in 38 out of 48 combinations, the backbones exhibited less than 5% deterioration in ROC AUC*, further underscoring the potential of our proposed backbone to not only simplify the graph structure but also to potentially uncover more salient features pertinent to the learning task.

Conversely, it is crucial to acknowledge instances where the proposed backbone representation led to a decrease in performance. The most significant reduction was observed with the NCI1 dataset, where the application of the GCN baseline model on the backbone graph saw a decline in ROC AUC by 10.68%. This suggests that while the proposed backbone can generally maintain or improve performance, there may be specific scenarios or datasets where the full topology of the original graph is necessary to capture the nuances required for better classification.

In Section 3, we introduced two methodologies for deriving control backbones from networks: the ZFS approach and the distance-based approach. Both methods are designed to ensure network controllability. Similar to B_z, the distance-based backbone, denoted as B_d [2], is crafted to maintain the lower controllability bound, preserving the network's control characteristics. We evaluated the effectiveness of these backbones by comparing their performance to the original graphs, and included random spanning trees, constructed using Kruskal's algorithm, as a baseline for learning and controllability. The empirical results, shown in Figure 4, reveal a clear trend: in most cases, the control backbones outperform the original graphs across various datasets and models. *In 67% of cases, the control backbones improve ROC AUC compared to the original graphs, with less than 5% deterioration in the remaining cases.* Additionally, *the control backbones outperform random spanning trees in approximately 80% of cases, with less than 2% deterioration in the rest.* These results demonstrate that the ZFS- and distance-based control backbones provide an effective solution for simplifying network structures while retaining essential control and learning properties.

5 Conclusion and Future Work

This work develops an effective sparse machine learning backbone for graphs using a ZFS-based approach. This method simplifies graph structures into tree-like forms while retaining essential control properties, enhancing learning efficiency. Extensive experiments demonstrate that the ZFS-based backbone not only preserves network controllability but often outperforms original graphs and other sparse representations in graph classification tasks. We also explored a distance-based backbone, showing its potential to generalize the controllability backbone and preserve critical characteristics across diverse networks. The ZFS-based backbone provides a robust, efficient solution for improving graph learning by simplifying structures



(a) Deezer Ego (b) Twitch Ego (c) Github Stargazers (d) Reddit Threads

(e) MUTAG (f) PTC (g) PROTEINS (h) NCI1

Fig. 4: Comparing the Efficacy of Network Backbone Structures for Graph Classification. The backbone represents the best-performing structure between B_z and B_d. The results are compared against the original graphs and random spanning tree subgraphs of the original graphs.

without sacrificing control attributes. Future research will refine the backbone computation process for large-scale applications and investigate the relationship between controllability and learning through the average degree of learning backbones.

6 Acknolwdgements

This work is supported by the National Science Foundation under Grant Numbers 2325416 and 2325417.

References

1. Abbas, W., Shabbir, M., Yazıcıoğlu, Y., Koutsoukos, X.: On zero forcing sets and network controllability–computation and edge augmentation. IEEE Transactions on Control of Network Systems (2023)
2. Ahmad, O.U., Shabbir, M., Abbas, W.: Controllability backbone in networks. In: 2023 62nd IEEE Conference on Decision and Control (CDC), pp. 2439–2444. IEEE (2023)
3. Ahmad, O.U., Shabbir, M., Abbas, W., Koutsoukos, X.: A graph machine learning framework to compute zero forcing sets in graphs. IEEE Transactions on Network Science and Engineering (2023)
4. AIM Minimum Rank Special Graphs Work Group: Zero forcing sets and the minimum rank of graphs. Linear Algebra and its Applications **428**(7), 1628–1648 (2008)
5. Barabasi, A.: Twenty years of network science: From structure to control. In: APS March Meeting Abstracts, vol. 2019, pp. S53–001 (2019)
6. Chan, H., Akoglu, L.: Optimizing network robustness by edge rewiring: a general framework. Data Mining and Knowledge Discovery **30**, 1395–1425 (2016)
7. Chen, T., Sui, Y., Chen, X., Zhang, A., Wang, Z.: A unified lottery ticket hypothesis for graph neural networks. In: International Conference on Machine Learning, pp. 1695–1706. PMLR (2021)
8. Gasteiger, J., Weißenberger, S., Günnemann, S.: Diffusion improves graph learning. Advances in Neural Information Processing Systems **32** (2019)

9. Godsil, C., Royle, G.F.: Algebraic graph theory, vol. 207. Springer Science & Business Media (2001)
10. Gu, S., Pasqualetti, F., Cieslak, M., Telesford, Q.K., Yu, A.B., Kahn, A.E., Medaglia, J.D., Vettel, J.M., Miller, M.B., Grafton, S.T., et al.: Controllability of structural brain networks. Nature communications **6**(1), 8414 (2015)
11. Hamidi, M., Chetouani, A., El Haziti, M., El Hassouni, M., Cherifi, H.: Blind robust 3d mesh watermarking based on mesh saliency and wavelet transform for copyright protection. Information **10**(2), 67 (2019)
12. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in Neural Information Processing Systems **30** (2017)
13. Karger, D.R.: Using randomized sparsi cation to approximate minimum cuts. In: Proc. 5th Symp. on Discrete Algorithms, vol. 424, p. 432 (1994)
14. Kruskal, J.B.: On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society **7**(1), 48–50 (1956)
15. Li, J., Shiu, W.C., Chang, A.: The number of spanning trees of a graph. Applied Mathematics Letters **23**(3), 286–290 (2010)
16. 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 (2020)
17. Pósfai, M., Liu, Y.Y., Slotine, J.J., Barabási, A.L.: Effect of correlations on network controllability. Scientific reports **3**(1), 1067 (2013)
18. Rital, S., Cherifi, H., Miguet, S.: Weighted adaptive neighborhood hypergraph partitioning for image segmentation. In: Pattern Recognition and Image Analysis: Third International Conference on Advances in Pattern Recognition, ICAPR 2005, Bath, UK, August 22-25, 2005, Proceedings, Part II 3, pp. 522–531. Springer (2005)
19. Rozemberczki, B., Kiss, O., Sarkar, R.: Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In: Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), p. 3125–3132. ACM (2020)
20. Said, A., Ahmad, O.U., Abbas, W., Shabbir, M., Koutsoukos, X.: Network controllability perspectives on graph representation. IEEE Transactions on Knowledge and Data Engineering (2023)
21. Said, A., Ahmad, O.U., Abbas, W., Shabbir, M., Koutsoukos, X.: Improving graph machine learning performance through feature augmentation based on network control theory. arXiv preprint arXiv:2405.03706 (2024)
22. Sontag, E.D.: Mathematical control theory: deterministic finite dimensional systems, vol. 6. Springer Science & Business Media (2013)
23. Topping, J., Di Giovanni, F., Chamberlain, B.P., Dong, X., Bronstein, M.M.: Understanding over-squashing and bottlenecks on graphs via curvature. arXiv preprint arXiv:2111.14522 (2021)
24. Tsitsulin, A., Perozzi, B.: The graph lottery ticket hypothesis: Finding sparse, informative graph structure. arXiv preprint arXiv:2312.04762 (2023)
25. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems **32**(1), 4–24 (2020)
26. Yassin, A., Cherifi, H., Seba, H., Togni, O.: A modular network exploration of backbone extraction techniques. In: International Conference on Complex Networks and Their Applications, pp. 296–308. Springer (2023)
27. Yazıcıoğlu, A., Abbas, W., Egerstedt, M.: Graph distances and controllability of networks. IEEE Transactions on Automatic Control **61**(12), 4125–4130 (2016)
28. Yazıcıoğlu, Y., Shabbir, M., Abbas, W., Koutsoukos, X.: Strong structural controllability of networks: Comparison of bounds using distances and zero forcing. Automatica **146**, 110,562 (2022)
29. Yu, J., Wang, N., Wang, G., Yu, D.: Connected dominating sets in wireless ad hoc and sensor networks–a comprehensive survey. Computer Communications **36**(2), 121–134 (2013)