Robust Deep Learning on Graphs

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Vassilis N. Ioannidis

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Prof. Georgios B. Giannakis, Advisor

August, 2020
Acknowledgments

First and foremost, my deepest gratitude goes to my advisor Prof. Georgios B. Giannakis. Professor gave me this amazing opportunity and welcomed me as his student in his prestigious research group. I am grateful for his mentoring that allowed me to mature personally and scientifically and his friendship that is invaluable for me. His vast knowledge, guidance and invaluable advice, have helped shape both this thesis and myself as an aspiring researcher.

Special thanks go to my thesis committee members, Prof. Mehmet Akcakaya, Prof. Mingyi Hong, Prof. George Karypis, and Prof. Zhi-Li Zhang for all their valuable comments and feedback on my research and thesis. I would also like to deeply thank Prof. Nikolaos Sidiropoulos, Prof. Antonio G. Marqués and Prof. Daniel Romero for our fruitful collaboration.

In addition, I am grateful to all of my professors at the University of Minnesota, whose graduate courses were enlightening and gave me the necessary tools for my research. Furthermore, for the fruitful discussions we have had, as well as their company I would like to thank all the current and previous members of the SPiNCOM group: Dr. Brian Baingana, Dr. Dimitris Berberidis, Dr. Elena Ceci, Dr. Jia Chen, Prof. Tianyi Chen, Georgios V. Karanikolas, Dr. Donghoon Lee, Prof. Geert Leus, Bingcong Li, Meng Ma, Dr. Athanasios Nikolakopoulos, Kostas Polyzos, Alireza Sadeghi, Dr. Fatemeh Sheikholeslami, Prof. Yanning Shen, Dr. Panagiotis Traganitis, Dr. Gang Wang, Dr. Yunlong Wang, Liang Zhang, Prof. Yu Zhang, Dr. Qin Lu, Seth Barrash; as well as my friends and colleagues from the Digital Technology Center and University of Minnesota: Charilaos Kanatsoulis, Agoritsa Polyzou, Ioanna Polyzou, Andreas Katis, Maria Kalantzi, Dr. Dimitrios Zermas, Dr. Ioannis Nompelis, Dr. Michail Vlysidis, Dr. Dionysios Angelidis, Dr. Athanasios Touris, Dr. Vasileios Kalantzis, Dr. Dimitrios Kalliotzis, Nikolaos Stefas, Nikos Memmos, Vasileios Charitatos, Lambros Tassoulas, Nikos Kargas, Dr. Mohit Sharma, Dr. Spyros Charonis, Vasileios Lytsakis, Dr. Shaden Smith, Saurav Manchanda, Dr. Evangelia Christakopoulou, Dr. Konstantina Christakopoulou, Bo Yang, Ahmed Zamzam, Costas
Mavromatis, and Faisal Almutairi. Let me thank my gym buddies Prof. Vassilios Morellas and Dr. Panagiotis Stanitsas. Their friendship, support and advice was instrumental in and out of the gym. I am also grateful to Theoni Greka for her support and friendship.

I am very grateful to my parents, Nikos and Evi and my sister Eleni. They supported me throughout my life and they gave me unconditional love, values, and all necessary opportunities to become the person I am today. My father guided me with his thoughtful advice to achieve my full potential.

Last but not least I am grateful to Elena for her support, companionship, advice and love. Meeting Elena was one of the most rewarding moments of the last 5 years.

Vassilis N. Ioannidis, Minneapolis, Summer, 2020.

The work in this Thesis was supported by NSF grants 1500713, 1514056, 1711471 and 1901134 and the Doctoral Dissertation Fellowship by the UMN.
This dissertation is dedicated to my family and Elena for all their love and support.
Abstract

The era of “data deluge” has sparked the interest in graph-based learning methods and their application in a number of disciplines ranging from sociology and biology to transportation or communications. Realizing the potential of graph-based learning has never been closer, even though formidable challenges are yet there to overcome. Contemporary graphs have massive scale up to billions of nodes, and generate unceasingly “big data”. Graph edges or node attributes may be only partially available due to application specific constraints, which calls for learning approaches to impute the missing information. Graph deep learning methods model complex nonlinear functions and achieve remarkable results in various tasks but the theoretical analysis of such methods is lacking. Last but not least, approaches to learning over graph data must be also robust to adversarial behavior. These challenges have been confronted only partly and separately under different formulations and application domains.

The proposed research is centered on analytical and algorithmic foundations that aspire to address the aforementioned challenges facing robust deep learning tasks over large-scale dynamic graphs. The overarching vision is to leverage and adapt state-of-the-art deep learning, optimization and networking tools for inference tasks based on limited graph data. Target applications include identifying node and edge anomalies, predicting node attributes, as well as providing graph-driven recommendations. The ultimate goal is to both analytically and numerically demonstrate how valuable insights from modeling graph data can lead to markedly improved learning tools.

To this end, the present thesis investigates three main research thrusts: i) unveiling anomalies on graphs; ii) robust deep learning on graphs; and iii) explaining deep learning on graphs via scattering transforms. The aforementioned research thrusts introduce novel methods that aim to tackle the challenges of robust deep learning on graphs. The potential of the proposed approaches is showcased by rigorous theoretical results and extensive experiments.
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Dataset characteristics</td>
<td>21</td>
</tr>
<tr>
<td>2.2</td>
<td>AUC values for detecting anomalous nodes under adversarial attacks</td>
<td>21</td>
</tr>
<tr>
<td>2.3</td>
<td>AUC values for detecting anomalous nodes under clusters of anomalous nodes</td>
<td>22</td>
</tr>
<tr>
<td>3.1</td>
<td>List of citation graph datasets considered in Secs. 3.5.2 and 3.5.3 along with most relevant dimensions</td>
<td>49</td>
</tr>
<tr>
<td>3.2</td>
<td>Classification accuracy for the setup described in Sec. 3.5.3 as the number of attacked nodes $</td>
<td>T</td>
</tr>
<tr>
<td>3.3</td>
<td>List of protein-to-protein interaction datasets considered in Sec. 3.5.4 and their associated dimensions</td>
<td>52</td>
</tr>
<tr>
<td>4.1</td>
<td>Dataset characteristics</td>
<td>73</td>
</tr>
<tr>
<td>4.2</td>
<td>Graph classification accuracy</td>
<td>74</td>
</tr>
<tr>
<td>4.3</td>
<td>SSL classification accuracy</td>
<td>76</td>
</tr>
</tbody>
</table>
List of Figures

1.1 Networks where proposed research is applied. ........................................... 2
1.2 Nodes in dotted square exhibit (a) (b) homophilic and (c) structural anomalies. . 3
1.3 A multi-relational network of voters. .......................................................... 4

2.1 An illustration of the operation of GraphSAC. The first row represents the available graph and labels. The second row shows the sampled labels. The third row contains the SSL module that outputs the predicted labels, where green dotted lines indicate incorrectly classified nodes. The filter in the fifth row decides whether $\mathcal{L}^{(r)}$ contains anomalies or not. The predictions with a larger number of misclassified nodes are discarded (colored red). The final anomalies are detected by combining the predictions from the “clean” subsets. .......................... 14
2.2 AUC values for increasing number of anomalies $|\mathcal{A}| = K$. (Top left) Pubmed, (Top middle left) Cora, (Top middle right) Citeseer, (Top right) Polblogs, (Bottom left) Blogcat, (Bottom middle) Wikipedia (Bottom right) PPI. ... 22
2.3 Runtime comparisons across graphs for Fig. 2.2. ........................................... 23
2.4 GraphSAC’s sensitivity to $R, \tau$ for the Citeseer dataset with $K = 80$ in Fig. 2.2 25
2.5 X-axis denotes the fraction of samples $S/N$ and y-axis represents the fraction of anomalies $K/N$. ................................................................. 25
2.6 Classification accuracy in Cora with $|\mathcal{M}| = 100$. ..................................... 26
2.7 Classification accuracy in Citeseer under anomalies. ...................................... 26
2.8 Classification accuracy in Polblogs under anomalies. .................................... 27
2.9 Classification accuracy in Cora under anomalies. ......................................... 29
2.10 Classification accuracy in Pubmed under anomalies. ................................. 30
3.1 The NAM combines the features using the multi-relational graph. The picture focuses on node $n$ and illustrates the case where $R$-hop neighbors (with $R = 2$) are considered. Note that, as shown in the picture, the local neighborhood is not the same across the different graphs.

3.2 The GAM combines the features per $i$, based on the trainable coefficients $\{R_{i/n}\}$.

3.3 TGCN with $L$ hidden (black) and one output (red) layers. The input $X$ contains a collection of features per node and the output to be predicted is the probability of each node to belong to each the $K$ classes (labels) considered. Each layer of the TGCN is composed of our three novel modules (NAM, GAM, FAM) described in equations (3.4), (3.5), and (3.6). Notice the skip connections that input $X$ to each layer [cf. (3.9)].

3.4 ED in operation on a perturbed social network among voters. Black solid edges are the true links and dashed red edges represent adversarially perturbed links.

3.5 Classification accuracy on the synthetic (a)-(b) and ionosphere (c)-(d) graphs described in Sec. 3.5.1 as the noise level in the features [cf. (3.17)] or in the links [3.16] varies. Panels (a) and (c) show the classification accuracy for noisy features while panels (b) and (d) show the same metric as the power of the noise added to the graph links varies.

3.6 Classification accuracy for the setup described in Sec. 3.5.2 as the number of perturbed edges increases.

3.7 SSL classification accuracy of the TGCN under varying edge creation prob. $q_1$, edge deletion prob. $q_2$, and number of samples $I$. 

3.8 Macro F1 values for different multi-relational networks.

4.1 Illustration of the same pGST applied to different graph datasets. For the social network (a) most GST branches are pruned, suggesting that most information is captured by local interactions.

4.2 Scattering pattern of a pGST with $J = 3$ and $L = 3$. Dashed lines represent pruned branches. An example $\Gamma$ and GFTs of filter banks are depicted too. The third filter $j = 3$ at $\ell = 1$ is pruned because it generates no output ($z_{(3)} = 0$).
4.3 Classification accuracy against number of samples in the authorship attribution (a) and SNR in dB for source localization (b). The percentage of features after pruning retained in the top-$2|T|$ most important GST features given by the SVM classifier (c).

4.4 Runtime comparison of the scattering transforms for Fig. 4.3.

4.5 Performance of pGSTs for varying $\tau$.

4.6 Classification accuracy over $L$.

4.7 Classification accuracy over $J$.

4.8 3D point cloud classification.

4.9 The pGST applied to the Cora dataset with and without signal perturbation, corroborating the result of Corollary 1.

4.10 The pGST applied to the Cora dataset with and without structural perturbations.

4.11 Illustration of the pGST applied to three protein datasets. Observe that after $l = 5$ most features are pruned.

4.12 Performance of DiffPool for different number of GCN layers.
Chapter 1

Introduction

A number of applications involve data that admit a natural representation in terms of node attributes over social, economic, sensor, communication, and biological graphs [45,59]. Prediction of neural signals from limited data over dynamic brain graphs may reduce the associated medical diagnosis costs. Mining social networks coupled with users’ purchase profiles facilitates personalized recommendations. Node classification in protein-protein interaction graphs associates proteins with specific biological functions, thereby facilitating the understanding of pathogenic and physiological mechanisms. Detecting suspicious activity in purchase networks uncovers fraudulent transactions. Identifying anomalous connections in political blogs may impede the propagation of “fake news.” Learning from these large volumes of graph data leads to significant science and engineering advances along with consequent improvements in quality of life. In the midst of “data deluge,” realizing this potential has never been closer, even though formidable challenges are yet there to overcome.

Contemporary graphs have massive scale up to billions of nodes, and generate unceasingly “big data” (60B messages, 3B likes, 350M pictures daily in Facebook). Interconnections change over time, which gives rise to dynamic graphs. This torrent of data necessitates on-the-fly (online) and scalable analytics. Graph edges or node attributes may be only partially available due to application specific constraints, which calls for learning approaches to impute the missing information. Nodal features may abide with nonlinear relations that requires judicious and expressive modeling. Often nodes are associated with large amounts of meta-information that requires methods tailored for multi-way (tensor) data. Nodes may be connected via multiple types or layers of relations such as those comprising multiple social ties among individuals in
family, friends, or coworker circles that gives rise to multi-layer graphs. Approaches to learning over graph data must be also robust to adversarial behavior. Contemporary graph deep learning methods achieve state-of-the-art performance in different tasks, nevertheless their theoretical analysis is lacking. These challenges have been confronted only partly and separately under different formulations and application domains.

This dissertation aspires to address the aforementioned challenges facing robust learning tasks over large-scale graphs by providing tangible answers, both theoretical and numerical, to the following practical questions Fig. 1.1.

Q1. Can we provide an anomaly detection framework that identifies suspicious activities or “fake” data in networked data?

Q2. Can we develop a robust deep learning framework for processing possibly multi-relational data?

Q3. Can we analyze theoretically the empirical success of graph deep learning models by exploring properties of their non-trainable counterpart the graph scattering transforms?

The ever-expanding interconnection of social, email, and media service platforms presents an opportunity for adversaries manipulating networked data to launch malicious attacks [2, 48, 100, 139]. Adversarially perturbed or simply anomalous graph data may disrupt the operation of critical machine learning algorithms with severe consequences. Detecting anomalies in graph data is of major importance in a number of contemporary applications such as flagging “fake news,” unveiling malicious users in social networks, blocking spamming users in email networks, and uncovering suspicious transactions in financial or e-commerce networks [3, 138]. Such anomalous data pose a serious risk for learning over graphs. Homophilic anomalies characterize nodes whose attributes are dissimilar to those of their neighbors [97, 101]. These nodes violate the homophily property that postulates neighboring vertices to have similar attributes, and is heavily
employed in semi-supervised learning (SSL) [46, 80, 82, 123]. In a social network of voters for example, friends typically belong to the same voting party; see Fig. 1.2a. Oftentimes, anomalous nodes may form dense connections giving rise to clustered homophilic anomalies Fig. 1.2b. Structural anomalies correspond to nodes with attributes that are dissimilar to structurally similar nodes [37]. Structural similarity among nodes suggests that vertices involved in similar graph structural patterns possess related attributes [36]. In an academic collaboration network for instance, nodes with similar graph structure (central nodes) have similar labels (e.g., professors); see Fig. 1.2c. Chapter 2 answers Q1 by designing an anomaly detection framework that can be integrated in different machine learning algorithms.

A task of major importance in the interplay between machine learning and network science is semi-supervised learning (SSL) over graphs. In a nutshell, SSL aims at predicting or extrapolating nodal attributes given: i) the values of those attributes at a subset of nodes and (possibly) ii) additional features at all nodes. A relevant example is protein-to-protein interaction networks, where the proteins (nodes) are associated with specific biological functions (the nodal attributes in this case are binary values indicating whether the protein participates in the function), thereby facilitating the understanding of pathogenic and physiological mechanisms. While significant progress has been achieved for this problem, most works consider that the relation among the nodal variables is represented by a single graph. This may be inadequate in many contemporary applications, where nodes may engage in multiple types of relations [81], motivating the generalization of traditional SSL approaches for single-relational graphs to multi-relational graphs; see Fig. 1.3. Many works in the literature refer to these graphs as multi-layer graphs [81]. In the particular case of protein interaction networks, each layer of the graph could correspond to a different type of tissue (brain, muscle...). Deep learning on graphs methods have achieved state-of-the-art performance in SSL. Graph convolutional networks (GCNs) postulate that the mapping between the input data and the labels is given by a neural network architecture.
that incorporates the graph structure and generalizes the typical convolution operations; see e.g., \cite{17, 41, 60, 80, 115, 122}. Chapter 5 develops a novel robust deep learning framework for SSL over\textit{ multi-relational} graphs.

GCNs and variants thereof have been remarkably successful in social network analysis, 3D point cloud processing, recommender systems and action recognition. However, researchers have recently reported less consistent perspectives on the desirable GCN designs. For example, experiments in social network analysis have argued that deeper GCNs marginally increase the learning performance \cite{132}, whereas a method for 3D point cloud segmentation achieves state-of-the-art performance with a 56-layer GCN network \cite{87}. These ‘controversial’ empirical findings motivate theoretical analysis to understand the fundamental performance-defining factors, and the resultant design choices for high-performance GCNs. Aiming to bestow GCNs with theoretical guarantees, one promising path is to study graph scattering transforms (GSTs) – an analysis framework that has been advocated to assess stability and explain the success of deep neural networks (DNNs) \cite{18, 93}. GSTs are non-trainable GCNs comprising a cascade of graph filter banks followed by nonlinear activation functions. The graph filter banks are designed analytically to scatter an input graph signal into multiple channels. GSTs extract stable features of graph data that can be utilized for downstream graph learning tasks \cite{43}, with competitive performance especially when the number of training examples is small. Under certain conditions on the graph filter banks, GSTs are endowed with energy conservation properties \cite{149}, as well as stability that amounts to robustness to graph topology deformations \cite{40}. Inherited from scattering transforms, GSTs however are known to incur exponential complexity in space and time that...
increases with the number of layers (GST depth) \cite{18, 93}. Furthermore, stability should not come at odds with sensitivity. A filter’s output should be sensitive to and able to cope with perturbations of large magnitude. Current GST efforts have not addressed transform sensitivity to input noise. Lastly, graph data in different domains (e.g., social networks versus 3D point clouds) have distinct properties, which prompts domain-adaptive GST designs. Chapter 4 addresses the aforementioned challenges and answers Q3.

1.1 Thesis Outline

The remainder of the thesis is organized as follows.

Chapter 2 of the present thesis introduces a graph random sampling and consensus (GraphSAC) framework for detecting anomalous nodes on large graphs. The novel method samples subsets of nodes, and relies on graph-aware criteria to judiciously discover anomalous nodes. GraphSAC is further utilized to enhance the performance of SSL algorithms, by detecting anomalies in subsets of nodes before predicting the labels of the rest. GraphSAC-SSL can be employed to complement contemporary deep learning SSL approaches as GCNs and GATs. The novel method scales to large-scale graphs since the complexity is linear in the number of edges. The proposed algorithms are evaluated with a rigorous performance analysis and extended numerical tests on real datasets. This work is included in \cite{55, 56, 58}.

Chapter 3 develops a deep learning framework for SSL over a collection of graphs with applications to both multi-relational data and robust learning. A tensor-GCN (TGCN) architecture is developed that accounts for multi-relational graphs. Learnable coefficients are introduced allowing the flexible model to adapt to the multiple graphs and identify the underlying structure of the data. The TGCN also facilitates robust SSL for single or multi-relational data when the underlying topology is perturbed. To defend against adversaries a novel edge-dithering (ED) approach is developed that generates ED graphs by sampling edges of the original graph with probabilities selected to enhance robustness. Numerical tests with multi-relational protein networks showcase the merits of the proposed tensor-graph framework. Further experiments under noisy features, noisy edge weights, and random as well as adversarial edge perturbations verify the robustness properties of the novel approach. This chapter is published in \cite{60, 71–74}.

Chapter 4 introduces a data-adaptive pruning approach to systematically retain informative features of the graph scattering transform (GST). The pruning decisions are guided by a criterion
promoting alignment (matching) of the input graph spectrum with that of the graph filters. The optimal pruning decisions are provided on-the-fly, and alleviate the exponential complexity of GSTs. The pruned-GST (pGST) is provably stable to perturbations of the input graph data and those of the network structure. Further, the sensitivity of pGST to random and localized noise is theoretically and experimentally investigated. The proposed pGSTs perform similar to, and in certain cases better than, the baseline GSTs that use all scattering features, while achieving significant computational savings. Furthermore, the efficient and stable features extracted can be utilized towards graph classification and 3D point cloud recognition. Even without any training on the feature extraction step, the performance is comparable to state-of-the-art deep supervised learning approaches, particularly when training data are scarce. The related publications can be found at [57, 70].

Finally, Chapter 5 presents a concluding discussion of the proposed approaches, along with future research directions.

## 1.2 Other works in my PhD

### 1.2.1 Coupled tensor and graph factorization

Joint analysis of data from multiple information repositories facilitates uncovering the underlying structure in heterogeneous datasets. Single and coupled matrix-tensor factorization (CMTF) has been widely used in this context for imputation-based recommendation from ratings, social network, and other user-item data. When this side information is in the form of item-item correlation matrices or graphs, existing CMTF algorithms may fall short. Alleviating current limitations, a novel model coined coupled graph-tensor factorization (CGTF) is introduced that judiciously accounts for graph-related side information. The CGTF model has the potential to overcome practical challenges, such as missing slabs from the tensor and/or missing rows/columns from the correlation matrices. A complementing direction is further explored by employing the interpretable factors to detect graph communities having the tensor as side information. The resulting community detection approach is successful even when some links in the graphs are missing. Results with real data sets corroborate the merits of the proposed methods relative to state-of-the-art competing factorization techniques in providing recommendations and detecting communities [68, 69, 144].
1.2.2 Online learning over dynamic graphs

Modeling vertex attributes as processes that take values over a graph allows for data processing tasks, such as filtering, inference, and compression, while accounting for information captured by the network topology [75, 83, 119]. However, if the topology is unavailable, inaccurate or even unrelated to the process of interest, performance of the associated task may degrade severely. For example, consider a social graph where the goal is to predict the salaries of all individuals given the salaries of some. Graph-based inference approaches that assume smoothness of the salary over the given graph, may fall short if the salary is dissimilar among friends. In various applications however, the network connectivity and node attributes change over time. We addressed such learning task with kernel-based approaches [59, 61, 67] for dynamic graphs [62–64, 109] and multi-relational graphs [89–92]. We also developed methods for joint learning of the network topology and dynamic processes [65, 66].

1.3 Notational Conventions

Scalars are denoted by lowercase, column vectors by bold lowercase, matrices by bold uppercase, and tensors by bold uppercase underscored letters. Superscripts $^\top$ and $^{-1}$ denote, respectively, the transpose and inverse operators; while $1_N$ stands for the $N \times 1$ all-one vector. Finally, if $A$ is a matrix and $x$ a vector, then $\|x\|_A^2 := x^\top A^{-1} x$ (provided that the inverse exists), $\|A\|_1$ denotes the $\ell_1$-norm of the vectorized matrix, and $\|A\|_F$ is the Frobenius norm of $A$. 
Chapter 2

GraphSAC: Robustifying learning and unveiling anomalies over graphs

The ever-expanding interconnection of social, email, and media service platforms presents an opportunity for adversaries manipulating networked data to launch malicious attacks [2, 48, 100, 139]. Adversarially perturbed or simply anomalous graph data may disrupt the operation of critical machine learning algorithms with severe consequences. Detecting anomalies in graph data is of major importance in a number of contemporary applications such as flagging “fake news,” unveiling malicious users in social networks, blocking spamming users in email networks, and uncovering suspicious transactions in financial or e-commerce networks [3, 138].

Such anomalous data pose a serious risk for semi-supervised learning (SSL) over graphs. SSL aims at predicting nodal labels given the graph connections and labels at a subset of nodes. This partial label availability may be attributed to privacy concerns (e.g., with medical data); energy considerations (e.g., with wireless sensor networks); or unrated items (e.g., with recommender systems). Standard SSL schemes typically assume that the available labels and graph connections have certain properties such as smoothness, which asserts that connected nodes have similar attributes [59, 123]. In various scenarios however, robustness issues arise. Powerful adversaries manipulate nodal attributes to bias learning, and promote their malicious goals [150]. Adversarially perturbed or simply anomalous graph data may degrade the performance of SSL algorithms with severe consequences. The recent era of misinformation and “fake” news calls for robust machine learning algorithms in network science [2, 48, 138]. Detecting these anomalous
nodes can be formulated as a learning task over an attributed graph.

2.1 Related Work

Today’s era of data deluge has grown the interest for detecting anomalies in collections of high-dimensional data [7, 10, 78, 113, 147]. This chapter deals with anomalies in data that exhibit inter-dependencies captured by a graph [3]. The inaccessibility and prohibitive cost associated with obtaining ground-truth anomalies motivates the development of mainly unsupervised techniques.

Methods for detecting anomalies in attributed graphs can be roughly classified in three categories. Community-based approaches find clusters of nodes and search for anomalies within each cluster. A probabilistic method is developed in [44] that jointly discovers communities, and detects community outliers as anomalies. Similarly, [101] identifies anomalies by measuring the attribute correlation of nodes within each node’s egonet, meaning the subgraph induced by the node of interest, its one-hop neighbors, and all their connections. Subspace-based approaches focus on spotting anomalies in subspaces extracted from the nodal features [114]. On the other hand, model-based methods learn an embedding per node and flag anomalies by measuring the model-fitting error [34, 88]. A parametric model is developed in [88] to capture the coherence among the attributes of nodes and their connectivity. A deep graph autoencoder is advocated in [34] that fuses attributes and connections to an embedding per node, and identifies anomalies using the reconstruction error at the decoder side. Despite their empirical success, these contemporary approaches are confronted with a number of challenges. The computational overhead associated with community detection, subspace extraction and deep learning, discourages their applicability to large-scale graphs. The local scope of community-based methods confines the breadth of the anomaly detector that is further vulnerable to clusters of connected anomalous nodes. Finally, all aforementioned approaches ultimately learn an anomaly score that relies on attributes and connections of all nodes. However, either the attributes or the network links for some nodes may be compromised by adversaries [136, 150].

Graph-based SSL methods typically assume that the true labels are “smooth” with respect to the underlying network structure, which naturally motivates leveraging the topology of the network to propagate the labels and increase classification performance. Graph-induced smoothness may be captured by kernels on graphs [12, 59, 123]; or Gaussian random fields [146].
Graph convolutional networks (GCN)s incorporate the graph structure and learn discriminating via optimizing the parameters of the network to achieve impressive results in SSL tasks \cite{17, 80, 137}. Graph attention networks (GAT)s introduce an attention mechanism to provide a tailored convolution operation that weights contributions from different nodes \cite{127}. With the success of GCNs on graph learning tasks granted, recent results indicate that perturbations of nodal labels can severely deteriorate their classification performance \cite{136, 150}. Detecting such graph anomalous data may unleash the potential of SSL over graphs.

### 2.1.1 Contributions

Addressing the aforementioned challenges, we introduce a graph random sampling and consensus (GraphSAC) framework for detecting anomalous nodes on large graphs. Instead of directly considering all nodes, our novel method samples subsets of nodes, and relies on graph-aware criteria to judiciously filter out subsets contaminated by anomalous nodes. The “clean” sets are utilized by a SSL module that estimates a nominal class distribution per node. The core intuition behind GraphSAC is that attributes of anomalous nodes will have poor predictive performance in a SSL task. GraphSAC is further utilized to enhance the performance of SSL algorithms, by detecting anomalies in subsets of nodes. Specifically, the contribution of this work is threefold.

**C1.** We put forth a novel approach that estimates a class distribution per node that is guaranteed to be minimally affected by anomalous nodes. The versatile anomaly detection framework adapts to different types of anomalies via an application-specific SSL module (cf. Fig. \ref{fig:framework}). The novel method scales to large-scale graphs since the complexity is linear in the number of edges.

**C2.** We adapt our GraphSAC approach to detect anomalous nodes over a subset of nodes and endow SSL approaches with increased robustness and enhanced prediction performance. GraphSAC-SSL can be employed to complement contemporary deep learning SSL approaches as GCNs and GATs.

**C3.** We showcase via extensive experiments that our novel graph anomaly detector outperforms state-of-the-art approaches in identifying clusters of anomalous nodes, as well as contemporary adversarial attacks on graph data. Additional experiments in SSL confirm the performance gains of employing GraphSAC towards robustifying SSL methods.
2.2 Graph Random Sampling and Consensus

Consider a graph $G := \{V, E\}$, where $V := \{n_1, n_2, \ldots, n_N\}$ is the vertex set, and $E$ the edge set of $F$ edges. The connectivity of $G$ is described by an adjacency matrix $A \in \mathbb{R}^{N \times N}$, where $A_{nn'} > 0$ if $(n, n') \in E$. Each node $n \in V$ is associated with one or more scalar labels $y_n \in \{1, \ldots, C\}$ that form the $N \times C$ matrix $Y := [y_1^\top, \ldots, y_N^\top]^\top$ with $Y_{nc} = 1$, if $y_n = c$, and 0 otherwise.

**Goal.** Given $A$ and $Y$ GraphSAC aims at detecting $K$ anomalous nodes with indices in the set $A := \{n_1, \ldots, n_K\}$. Such nodes are expected to violate a certain property such as homophily. To this end, we require a model that relates the graph with the labels, and promotes the desired properties.

An immediate approach is to directly consider all nodal labels and connections in a graph-based model. However, such a holistic approach will be vulnerable to the inclusion of anomalous nodes that will bias the learned model and poison the anomaly detection framework.

Instead, our fresh idea is to sample labels $y_n$ at random subsets of nodes $n \in L \subset V$ and prudently discard contaminated subsets.

Given $L$ and $A$, we rely on SSL (e.g., variants of $k-$ nearest neighbor rule [27]) to predict labels of nodes $n \in U$ with $U := V \setminus L$. SSL methods typically estimate a $N \times C$ label probability mass function matrix (pmf)

$$\hat{P} := f(\{y_n\}_{n \in L}, A)$$

(2.1)

where $\hat{P}_{nc} \in [0, 1]$ represents probability that $y_n = c$ and $f(\cdot, \cdot)$ is the SSL function. The choice of the SSL algorithm is dictated by complexity constraints and specific properties one may want to capture; see also Fig. 1.2.

Nevertheless, if $L \cap A \neq \emptyset$, the predicted label distributions will be affected by the anomalous nodes. To bypass this hurdle, we test to assess if anomalies are present in $L$ by evaluating the predictive SSL performance instantiated with $L$. Our test relies on the premise that attributes of anomalous nodes will have poor predictive performance for SSL.

Our novel algorithm is termed graph sample and consensus (GraphSAC)\(^1\) and utilizes multiple drawn samples executing per draw the following three steps.

\(^1\)We draw inspiration from the random sampling and consensus approach for robust model fitting in image analysis [14].
\textbf{S1.} First, }L \leq N\text{ nodes are sampled uniformly at random forming the set }L^{(r)} \in \{ \mathcal{L} \subset \mathcal{V}: |\mathcal{L}| = L \}

\textbf{S2.} Given the labels in }L^{(r)}\text{ and }\mathbf{A},\text{ the chosen SSL model estimates }\hat{\mathbf{P}}^{(r)}\text{. Nodes with correctly predicted labels form the consensus set }\mathcal{C}^{(r)} := \{ n \in \mathcal{U}^{(r)} : y_n = \hat{y}_n^{(r)} = \arg \max_c \hat{P}_{nc}^{(r)} \}\text{.}

\textbf{S3.} The accuracy of }\hat{\mathbf{P}}^{(r)}\text{ is evaluated by comparing the cardinality of the consensus set }|\mathcal{C}^{(r)}|\text{ to a threshold }\tau.\text{ If }|\mathcal{C}^{(r)}| > \tau N,\text{ GraphSAC decides that }L^{(r)}\text{ does not contain anomalies and sets an indicator }b^{(r)} = 1,\text{ otherwise, the set is considered contaminated with anomalies and }b^{(r)} = 0.\text{ After }R\text{ parallel executed draws, the sample average of the pmf matrices is given by}

\[
\hat{\mathbf{P}} = \frac{\sum_{r=1}^{R} \hat{\mathbf{P}}^{(r)} b^{(r)}}{\sum_{r=1}^{R} b^{(r)}}.\tag{2.2}
\]

Given }\hat{\mathbf{P}},\text{ GraphSAC estimates an }N \times 1\text{ anomaly score vector }\mathbf{s}\text{ with the }n\text{th entry}

\[
s_n = \text{dist}(\hat{\mathbf{p}}_n, \mathbf{y}_n)\tag{2.3}
\]

where }\mathbf{y}_n\text{ is the }n\text{-th row of }\mathbf{Y}, \hat{\mathbf{p}}_n\text{ is the }n\text{-th row of }\hat{\mathbf{P}},\text{ and }\text{dist}(\cdot, \cdot)\text{ is the cross-entropy loss. Therefore, }s_n\text{ is larger if }n\text{ does not adhere to the graph-related properties promoted by the SSL model. Finally, the }K\text{ nodes with the largest anomaly scores are declared as anomalous.}

Even though }L^{(r)}\text{ are drawn at random in S1 and may contain anomalous nodes, S3 introduces a sampling bias towards “clean” subsets. With S3, we aspire to filter out subsets that are contaminated with anomalies and will bias the learned pmf matrix. Our test relies on the premise that attributes of anomalous nodes will have poor predictive performance for SSL. Consequently, }\hat{\mathbf{P}}\text{ is minimally affected by anomalous nodes and estimates the nominal class distribution. Instead of using as many nodes as possible to obtain the solution, the fresh look advocated by GraphSAC is to rely on small sets of }L\text{ nodes and SSL-aided tests to avoid subsets contaminated with anomalous nodes. The small sample size } (L \ll N)\text{ enables GraphSAC to remain operational even under adverse conditions where }K\text{ is relative large. The computational complexity of GraphSAC per }r\text{ is linear with the number of edges for scalable SSL methods }[25][146].\text{ Further, since the sampled sets }L^{(r)}\text{ are independent, each draw }r\text{ can be readily parallelized, thereby ensuring scalability to large-scale graphs.}

A major departure from traditional ADoG literature, is that the novel AD framework is not restricted to a specific SSL model. GraphSAC may adapt to the pertinent type of anomalies by
Algorithm 2.1 GraphSAC

\begin{algorithmic}
\State \textbf{Input:} $f(\cdot, \cdot), A, \{y_n\}_{n=1}^N, R, \tau, r \leftarrow 0$
\State 1. \textbf{while} $r < R$ \textbf{do}
\State 2. \quad Select $L^{(r)}$ at random
\State 3. \quad Estimate $\hat{P}^{(r)}$ and find consensus set $C^{(r)}$
\State 4. \quad If $|C^{(r)}|/N < \tau$ then $b^{(r)} = 0$, otw. $b^{(r)} = 1$
\State 5. \quad $r \leftarrow r + 1$
\State 6. \textbf{end while}
\State 7. Obtain $\hat{P}$ as in (2.2)
\State 8. Obtain anomaly scores $\{s_n\}_{n=1}^N$ as in (2.3)
\end{algorithmic}

appropriately choosing the SSL model. Homophilic anomalies for example, call for SSL methods e.g. diffusion-based classifiers [13, 146] or contemporary graph convolutional neural networks (GCN)s [80]. On the other hand, structural anomalies necessitate models that promote structural similarities among nodes such as the work in [36]. Colluding adversaries, such as terrorist cells, may collaborate to deceive contemporary detection mechanisms. These collaborations may result in clustered anomalies. In addition, detecting these clustered anomalous nodes involves additional challenges since these node may disguise their identity. Experiments show that our GraphSAC can successfully detect such clustered anomalies, since our sampling technique is not affected by these.

GraphSAC vs RanSAC. It is instructive to highlight key differences between the novel GraphSAC and RanSAC [14, 28, 102, 106]. RanSAC is mainly established in computer vision for fitting robust models with the data defined on regular grids (images). On the other hand, GraphSAC fits SSL models with data defined over a graph, and the models are directly affected by the graph topology. Instead of building a sample average model minimally affected by ‘contaminated’ draws as in (2.2), RanSAC pursues a single model that contains no outliers. However, the latter approach is not effective for graph data, since the location of the perturbed node in the graph affects the SSL model more prominently than the location of an outlier in an image. RanSAC’s simple yet versatile design allows for rigorous analysis that assumes knowledge over the probability that a point is an outlier. Because of the aforementioned distinctions, our GraphSAC requires a tailored approach to establish analytical guarantees.
Figure 2.1: An illustration of the operation of GraphSAC. The first row represents the available graph and labels. The second row shows the sampled labels. The third row contains the SSL module that outputs the predicted labels, where green dotted lines indicate incorrectly classified nodes. The filter in the fifth row decides whether $L^{(r)}$ contains anomalies or not. The predictions with a larger number of misclassified nodes are discarded (colored red). The final anomalies are detected by combining the predictions from the “clean” subsets.
2.3 Analytical guarantees

This section strengthens our randomized anomaly detection framework with analytical guarantees. Towards streamlining the analysis, we model S3 as introducing a sampling bias to the uniform sampling scheme towards “clean” subsets, which are subsets that do not contain anomalous nodes. Specifically, (2.2) is rewritten as

\[ \hat{P} := \frac{1}{R} \sum_{r=1}^{R} \hat{P}^{(r)} \]  

(2.4)

where \( L^{(r)} \sim p_{G}(L) \) are drawn from the GraphSAC biased sampling scheme with

\[
p_{G}(L) = \begin{cases} 
    p_{d}(L), & \text{for } L \cap A = \emptyset \\
    p_{f}(L), & \text{for } L \cap A \neq \emptyset 
\end{cases}
\]  

(2.5)

where \(|L| = L, p_{d}(L)\) is the probability of sampling a clean subset \( L \), and \( p_{f}(L) \) the probability of sampling a subset containing at least one anomalous node. The pmf in (2.5) is related to the indicator variable \( b \) in S3 as follows

\[
p_{d}(L) = \Pr(b = 1 | L \in \tilde{L}_{L}) \Pr(L \in \tilde{L}_{L}) 
\]

(2.6)

\[
p_{f}(L) = \Pr(b = 1 | L \in \tilde{L}_{L}^{c}) \Pr(L \in \tilde{L}_{L}^{c})
\]

(2.7)

where \( \Pr(L \in \tilde{L}_{L}) = |\tilde{L}_{L}|/|L_{L}| \) and \( \Pr(L \in \tilde{L}_{L}^{c}) = |\tilde{L}_{L}^{c}|/|L_{L}| \) with \( \tilde{L}_{L} := \{ \mathcal{L} \subseteq V, |\mathcal{L}| = L, \mathcal{L} \cap A = \emptyset \} \) the set containing subsets without anomalous nodes, while the complementary set \( \tilde{L}_{L}^{c} \) contains all the remaining size-\( L \) subsets \( \tilde{L}_{L}^{c} := L_{L} \setminus \tilde{L}_{L} \). Notice that in (2.4) different than (2.2) all \( R \) samples are included in the sample average (2.4), but if a sample \( r \) is contaminated, i.e. \( L^{(r)} \in L_{L}^{c}, \mathcal{L}^{(r)} \) will have a smaller probability to be sampled.

GraphSAC aims at a class distribution per node that is not affected by the anomalous nodes, but takes into account only the nominal nodes, and can thus be readily utilized for anomaly detection. Hence, the desired probability matrix is

\[
P_{\text{nom}} := E[L^{(r)} \cap A = \emptyset | \hat{P}^{(r)}]
\]

(2.7)

where \( L^{(r)} \) are drawn uniformly from the set of nodal subsets without anomalous nodes. Indeed, \( P_{\text{nom}} \) captures the nominal class distribution per node that is not affected by the anomalous nodes.
and conforms to the properties promoted by the SSL model. As a result, the largest entries in $s$ (cf. (2.3)) indicate the anomalous nodes.

However, direct estimation of $P_{\text{nom}}$ is not feasible since $A$ is unknown. If on the other hand all nodes are directly accounted for, anomalous ones will be included that will in turn bias the learned probability matrix. To obviate this hurdle, GraphSAC introduces S3 to filter out contaminated subsets by appropriately setting the indicator variable $b$. GraphSAC’s ultimate target is the expected pmf matrix

$$P := \mathbb{E}_{L^{(r)} \sim p_G}[\hat{P}^{(r)}].$$  \hspace{1cm} (2.8)

where $L^{(r)}$ are drawn from the GraphSAC’s biased sampling scheme in (2.5). Upon considering $p_f(L) = 0$ for $L \cap A \neq \emptyset$ and $p_d(L) = 1/|\hat{L}|_L$ for $L \cap A \neq \emptyset$, as well as expanding the expectation in (2.8) and (2.7), GraphSAC’s expected probability matrix reduces to the desired one, meaning $P = P_{\text{nom}}$. This corresponds to a perfect filter in S3 that disregards all contaminated subsets $b^{(r)} = 0$ if $L^{(r)} \cap A \neq \emptyset$, and retains all the clean ones $b^{(r)} = 1$ if $L^{(r)} \cap A = \emptyset$.

Thus, GraphSAC’s performance is directly related to the distance among the estimated $\hat{P}$ and the nominal $P_{\text{nom}}$

$$\|\hat{P} - P_{\text{nom}}\|_1 \leq \|P - P_{\text{nom}}\|_1 + \|\hat{P} - P\|_1$$  \hspace{1cm} (2.9)

where (2.9) follows from the triangle inequality after adding and subtracting $P$, and $\|Z\|_1$ represents the sum of the absolute values of the $Z$ entries. Adhering to (2.9), the following research questions have to be addressed to characterize the performance of GraphSAC.

**RQ1.** How $\|P - P_{\text{nom}}\|_1$ relates to the filter performance? and

**RQ2.** How $\|\hat{P} - P\|_1$ evolves as the number of draws $R$ increases?

The following analysis aspires to provide tangible answers to aforementioned questions. First, consider a simplified version of $p_G(L)$ given by

$$p_G(L) = \begin{cases} d, & \text{for } L \cap A = \emptyset \\ f, & \text{for } L \cap A \neq \emptyset . \end{cases}$$  \hspace{1cm} (2.10)
This approximation suggests that GraphSAC samples all contaminated samples with a certain probability, and all non-contaminated with a different one (cf. (2.6)).

**Theorem 2.1.** (Proof in Sec. A.1) Let $P_{anom} := E_{L \cap A = \emptyset} [\hat{P}^{(r)}]$ denote the expected pmf matrix when anomalies are sampled. It then holds that

$$
\|P - P_{nom}\|_1 = \frac{|\hat{L}_E^c|^2}{|\hat{L}_E|} P_{fa} \|P_{anom} - P_{nom}\|_1
$$

(2.11)

$$
= |\hat{L}_E^c| f \|P_{anom} - P_{nom}\|_1
$$

(2.12)

where $P_{fa} := Pr(b^{(r)} = 1|L^{(r)} \cap A \neq \emptyset)$ is the probability of false alarms for S3 that can be also expressed as $P_{fa} = (|L_L|/|\hat{L}_E^c|)f$ with $|\hat{L}_E^c| = \binom{N}{L} - \binom{N-K}{L}$ and $|L_L| = \binom{N}{L}$.

Theorem 1 asserts that the desired distance is equal to the distance between the nominal distribution and the one affected by the anomalies scaled by the probability that GraphSAC fails to identify the anomalies. An immediate observation is that for a perfect GraphSAC filter it holds that $P_{fa} = 0$, and one would obtain the desired $P = P_{nom}$. Another consequence of Theorem 1, is that as the number of anomalies $K$ increases, so does $|\hat{L}_E^c|$ and $\|P - P_{nom}\|_1$, conditioned that the filter performance does not change. As demonstrated in the experiments, even if anomalous subsets are miss-classified as nominal by the filter for a specific draw, these will contain a small number of anomalies and will not affect the overall anomaly detection performance. On the other hand, if a subset contains a large number of anomalies, the probability this sample is rejected is higher relative to a sample with only a few anomalies. The following theorem formalizes these observations to achieve stronger results compared to Theorem 1.

**Theorem 2.2.** (Proof in Sec. A.2) Consider that ‘clean’ subsets will be retained, $Pr(b^{(r)} = 1|L^{(r)} \cap A = \emptyset) = 1$, and that subsets containing more than $K_m$ anomalies will be discarded $Pr(b^{(r)} = 0|L^{(r)} \cap A > K_m) = 1$. It holds that

$$
\|P - P_{nom}\|_1 = P_{fa} P_{L_K} P_{L_{K_m}} - P_{nom}(1 - P_{L_S})\|_1.
$$

(2.13)

where $P_{L_K} := Pr(L \in L^K_S)$ and $P_{L_S} := Pr(L \in L_S)$ are sampling probabilities, and $L^K_S := \{L : L \subset V, |L \cap A| \leq K_m\}$ and $L_S := \{L : L \subset L_S, L \cap A = \emptyset\}$ are the sets containing nodal subsets with at most $K_m$ and 0 anomalies respectively.
Different than Theorem 1, Theorem 2, utilizes two additional assumptions, and provides a finer representation of \( \| \mathbf{P}_G - \mathbf{P}_{\text{nom}} \|_1 \). Both of these assumptions are experimentally verified in Sec. 2.5.1.

Towards addressing Q2 we utilize concentration inequalities theory applied to random matrices [124] and establish the following.

**Theorem 2.3.** *(Proof in Sec. A.3.)* With \( \hat{\mathbf{P}} = \frac{1}{R} \sum_{r=1}^{R} \mathbf{P}^{(r)} \) and \( \mathbf{P} = \mathbb{E}_{\mathcal{L}^{(r)} \sim p_{\mathcal{G}}} [ \hat{\mathbf{P}}^{(r)} ] \), it holds that

\[
\mathbb{E}\{ \| \hat{\mathbf{P}} - \mathbf{P} \| \} \leq \sqrt{\frac{2N \log(N + C)}{R}} + \frac{2\sqrt{N \log(N + C)}}{3R} \tag{2.14}
\]

and for all \( t > 0 \)

\[
\mathbb{P}\{ \| \hat{\mathbf{P}} - \mathbf{P} \| \geq t \} \leq (N + C) \exp \left( \frac{-Rt^2}{N + 2\sqrt{N}t/3} \right). \tag{2.15}
\]

All the parameters in the right hand side of (2.14) and (2.15) are either known or controllable, and relate to the network size \( N \) and the number of classes \( C \). The expectation bound (2.14) suggests that by increasing the number of draws \( R \) the sample average of the random matrices will approach the desired ensemble mean. On the other hand, (2.15) bounds the tail of the probability distribution of the difference. It can be shown that for \( t \leq \sqrt{N} \) the tail probability decays as fast as the tail of a Gaussian random variable with variance proportional to \( N \), while for \( t \geq \sqrt{N} \) it decays as that of an exponential random variable whose mean is proportional to \( \sqrt{N} \).

Retracing back to (2.9), we deduce that \( \| \hat{\mathbf{P}} - \mathbf{P} \|_1 \) decreases for increasing \( R \), whereas \( \| \mathbf{P}_{\text{nom}} - \mathbf{P} \|_1 \) directly depends on the filter performance. In practice, GraphSAC amounts to ranking the per-node anomaly scores. This ranking may be evident even for relative large distances \( \| \hat{\mathbf{P}} - \mathbf{P}_{\text{nom}} \|_1 \).

### 2.3.1 Analysis on diffusion-based SSL models

The results presented so far hold for any SSL model even for contemporary GCNs. Next, we focus on the class of diffusion-based SSL models, which have documented success in sizable graphs due to their scalability and robustness [25, 146]. Most of these models can be written as

\[
\hat{\mathbf{P}}^{(r)} = h(A) \mathbf{Y}_{\mathcal{L}^{(r)}} \tag{2.16}
\]
Algorithm 2.2 GraphSAC-SSL

**Input:** \( f(\cdot, \cdot), g(\cdot, \cdot), A, \{y_n\}_{n \in M} \)

1. \( \{\phi_n\}_{n \in M} := \text{GRAPHSAC}(f(\cdot, \cdot), A, \{y_n\}_{n \in M}) \)
2. Select \( M_c \) nodes with smallest \( \{\phi_n\}_{n \in M} \), \( M_c \subset M \)
3. \( \hat{P} = g(\{y_n\}_{n \in M_c}, A) \)
4. Obtain labels \( y_n = \arg \max_{c} \hat{P}_{nc}, n \in U \)

where \( h \) is e.g. a polynomial function, and \( Y_L \) is an \( N \times C \) matrix whose \( n \)-th row is \( y_n \) if \( n \in L \), and \( 0_C \) otherwise. A common choice is
\[
h(A) = \sum_{t=0}^{T} \alpha_t (D^{-1/2} A D^{-1/2})^t,
\]
where \( \alpha_t > 0 \) and \( D = \text{diag}(A 1) \) denotes the degree matrix.

**Corollary 2.1.** (Proof in Section C of the supplementary material.) Let \( h(A) = [h_1, \ldots, h_{\text{nom}}] \) be the diffusion matrix, and \( \mathcal{N} := V - A \) the set containing the nominal nodes. The total variation distance between \( P \) and \( P_{\text{nom}} \) is
\[
\|P - P_{\text{nom}}\|_1 = P_{fa} \frac{f_A}{|L^c|} \left\| K \sum_{n \in \mathcal{N}} h_n y_n^T - \sum_{n' \in A} h_{n'} y_{n'}^T \right\|_1 \tag{2.17}
\]

where \( f_A := \binom{N-1}{L-1} \) and \( |L^c| = \binom{N}{L} - \binom{N-K}{L} \).

Upon applying the reverse triangle inequality to (2.17), Corollary 1 yields
\[
\|P_{\text{nom}} - P_{\text{anom}}\|_1 \geq \tag{2.18}
\]
\[
\geq \frac{f_A}{|L^c|} \left\| \sum_{n' \in A} h_{n'} y_{n'}^T \right\|_1 - \frac{K}{N-K} \left\| \sum_{n \in \mathcal{N}} h_n y_n^T \right\|_1 \]
\[
= \frac{f_A}{|L^c|} \left\| \sum_{n' \in A} h_{n'} \right\|_1 - \frac{K}{N-K} \sum_{n \in \mathcal{N}} \|h_n\|_1 \tag{2.19}
\]

where (2.19) follows since \( y_n \) has entries either 0 or 1, and \( h_n \) has nonnegative entries. Hence, anomalous nodes with large \( \|h_n\|_1 \) contribute more to the error norm \( \|P_{\text{nom}} - P\|_1 \). In diffusion-based models, nodes with larger \( \|h_n\|_1 \) typically have higher degree. Consequently, as expected anomalous nodes with high degree have a large effect on the learned distributions.
2.4 Robust SSL via GraphSAC

This section addresses the challenging task of semi-supervised learning (SSL) over graphs in the presence of perturbations of the sampled labels. Toward this end, the novel GraphSAC will be employed to enhance and robustify the learning performance.

**Goal.** Given the topology $A$, and labels only at a subset $M$ of nodes $\{y_n\}_{n \in M}$ with $M \subset V$, SSL over graphs aims at predicting the labels of nodes in the unlabeled set $\{y_n\}_{n \in U}$ with $U := V - M$. The additional challenge addressed in this section is that a subset of the available nodal labels are perturbed, i.e. $A \subset M$.

An immediate approach towards predicting the unavailable labels is to employ a SSL function $g(\cdot, \cdot)$ as follows

$$\hat{P} = g(\{y_n\}_{n \in M}, A)$$

(2.20)

where $\hat{P}$ holds the predicted label distributions. However, such a SSL scheme (2.20) will propagate the anomalous labels in $A$ and degrade the overall learning performance.

Our idea is to utilize the novel GraphSAC to identify anomalous nodes in the sampled set $M$. In Section 2.2 GraphSAC deliberately samples subsets of nodes $L \subset V$ given all labels to obtain the anomaly scores. On the other hand here, since only a subset $M$ of labels is available, GraphSAC samples subsets within the set of given nodes $L \subset M$. Hence, GraphSAC is applied to $M$ and the anomaly scores $\{\phi_n\}_{n \in M}$ are obtained, which can be ranked to unveil the anomalous nodes. The $M_c$ nodes with the smallest $\phi_n$ are selected as the ‘clean’ labels $M_c \subset M$. Subsequently, the SSL function $g(\cdot, \cdot)$ utilizes only the nodes in $M_c$ as follows

$$\hat{P} = g(\{y_n\}_{n \in M_c}, A).$$

(2.21)

Finally, the predicted labels are obtained as $y_n = \arg \max_c \hat{P}_{nc}, n \in U$. By utilizing the labels only at the clean nodal subset $M_c$, our robust SSL framework aspires to avoid the perturbed labels from contaminating the predicted labels. Remarkably, although the number of nodes used for SSL in (2.21) is less compared to (2.20), i.e. $|M_c| < |M|$, the learning performance is enhanced.

The SSL model $g(\cdot, \cdot)$ employed in (2.21) for prediction may be different than $f(\cdot, \cdot)$ used in S2 by the GraphSAC module for unveiling the anomalies. This decoupling between the
Table 2.1: Dataset characteristics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Nodes</th>
<th>Classes</th>
<th>Multilabeled nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>2708</td>
<td>7</td>
<td>No</td>
</tr>
<tr>
<td>Citeseer</td>
<td>3327</td>
<td>6</td>
<td>No</td>
</tr>
<tr>
<td>Pubmed</td>
<td>19717</td>
<td>3</td>
<td>No</td>
</tr>
<tr>
<td>Polblogs</td>
<td>1224</td>
<td>2</td>
<td>No</td>
</tr>
<tr>
<td>Blogcat</td>
<td>10312</td>
<td>39</td>
<td>Yes</td>
</tr>
<tr>
<td>PPI</td>
<td>3890</td>
<td>50</td>
<td>Yes</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>4733</td>
<td>39</td>
<td>Yes</td>
</tr>
</tbody>
</table>

prediction and anomaly discovery step endows our framework with increased flexibility. For example, an efficient diffusion-based SSL [46] can be employed in S2 to unveil anomalies, while a state-of-the-art but computational intensive SSL model, e.g. a GAT [127], can be utilized for $g(\cdot, \cdot)$ to obtain accurate predictions.

Notice that the exact number of anomalous nodes is not assumed known. If such prior knowledge was available then an appropriate design for the size of the retained subset would be $|\mathcal{M}_c| = |\mathcal{M}| - K$. Extensive experiments demonstrate the effect of $|\mathcal{M}_c|$ to the overall classification accuracy in Sec. 2.5.2.

Table 2.2: AUC values for detecting anomalous nodes under adversarial attacks.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Citeseer</th>
<th>Polblogs</th>
<th>Cora</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphSAC</td>
<td>0.75</td>
<td>0.98</td>
<td>0.80</td>
<td>0.82</td>
</tr>
<tr>
<td>Gae</td>
<td>0.64</td>
<td>0.51</td>
<td>0.50</td>
<td>0.69</td>
</tr>
<tr>
<td>Amen</td>
<td>0.73</td>
<td>0.89</td>
<td>0.75</td>
<td>0.62</td>
</tr>
<tr>
<td>Radar</td>
<td>0.67</td>
<td>0.76</td>
<td>0.77</td>
<td>0.44</td>
</tr>
<tr>
<td>Degree</td>
<td>0.58</td>
<td>0.48</td>
<td>0.40</td>
<td>0.57</td>
</tr>
<tr>
<td>Cut ratio</td>
<td>0.49</td>
<td>0.51</td>
<td>0.35</td>
<td>0.55</td>
</tr>
<tr>
<td>Flake</td>
<td>0.47</td>
<td>0.61</td>
<td>0.46</td>
<td>0.60</td>
</tr>
<tr>
<td>Conductance</td>
<td>0.35</td>
<td>0.39</td>
<td>0.61</td>
<td>0.59</td>
</tr>
</tbody>
</table>

2.5 Experiments

In this section, we evaluate GraphSAC’s detection performance under different anomaly generation models based on random walks, clustered anomalies, as well as contemporary adversarial
Table 2.3: AUC values for detecting anomalous nodes under clusters of anomalous nodes.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Citeseer</th>
<th>Polblogs</th>
<th>Cora</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphSAC</td>
<td>0.88</td>
<td>0.74</td>
<td>0.97</td>
<td>0.96</td>
</tr>
<tr>
<td>Gae</td>
<td>0.50</td>
<td>0.30</td>
<td>0.58</td>
<td>0.95</td>
</tr>
<tr>
<td>Amen</td>
<td>0.51</td>
<td>0.10</td>
<td>0.48</td>
<td>0.42</td>
</tr>
<tr>
<td>Radar</td>
<td>0.86</td>
<td>0.40</td>
<td>0.88</td>
<td>0.88</td>
</tr>
<tr>
<td>Degree</td>
<td>0.46</td>
<td>0.10</td>
<td>0.45</td>
<td>0.29</td>
</tr>
<tr>
<td>Cut ratio</td>
<td>0.49</td>
<td>0.08</td>
<td>0.40</td>
<td>0.27</td>
</tr>
<tr>
<td>Flake</td>
<td>0.40</td>
<td>0.05</td>
<td>0.37</td>
<td>0.33</td>
</tr>
<tr>
<td>Conductance</td>
<td>0.55</td>
<td>0.55</td>
<td>0.40</td>
<td>0.37</td>
</tr>
</tbody>
</table>

Figure 2.2: AUC values for increasing number of anomalies $|\mathcal{A}| = K$. (Top left) Pubmed, (Top middle left) Cora, (Top middle right) Citeseer, (Top right) Polblogs, (Bottom left) Blogcat, (Bottom middle) Wikipedia (Bottom right) PPI.

attacks for graph data. Furthermore, we assess the performance gains of GraphSAC in SSL over graphs, when anomalies are present in the sampled set. The goal here is to provide tangible answers to the following research questions.

**RQ1.** How does GraphSAC compare to state-of-the-art methods for anomaly detection on graphs under various anomaly generation models?

**RQ2.** How significantly GraphSAC enhances the performance of contemporary SSL models when anomalies are present in the sampled set?

### 2.5.1 Anomaly detection on graphs

**Baselines.** Amen identifies anomalies by evaluating the attribute correlation of nodes per egonet of the graph [101]. The graph neural network encoder (GAE) ranks anomalies by measuring...
the reconstruction error recovered at the decoder output [34]. Radar asserts nodes as anomalous if they do not adhere to the proposed parametric model [88]. Following the experimental setup in [101], additional anomaly detection methods are considered that only utilize graph connectivity but not nodal labels, namely approaches based on the Average degree [26], Cut ratio [39], Flake [38], and Conductance [4]. Unless stated otherwise, GraphSAC is configured with $\tau = 0.5$, $R = 50$ and the personalized PageRank (PPR) [5] as the SSL model. The different methods are evaluated using the area under the curve (AUC) of the receiver operating characteristic (ROC) curve. The ROC curve plots the rate an anomaly is detected (true positive) against the rate a node is miss-classified as anomalous (false positive). The AUC value represents the probability that a randomly chosen abnormal node is ranked higher than a normal node.

Datasets. The 7 benchmark labeled graphs used in the following experiments are included in Table 2.1. The nodes in the last three graphs are multilabeled ones.

Adversarial attacks

We generated anomalies using the adversarial setup in [150], where attacks are effected on attributed graphs targeted for GCNs. We focus on structural attacks, which means that edges adjacent to the targeted node are added or removed; that is, we select a random subset of targeted nodes $A$, and alter their connectivity by a sequence of structural attacks [150].

Table 2.2 reports the AUC values for competing state-of-the-art techniques in detecting adversarial attacks with $K=10$ targeted nodes. As GAE relies on a deep graph autonencoder [34], it is maximally affected by the adversarial attacks. Our novel method outperforms all alternatives in detecting the attacked nodes. These promising results suggest that GraphSAC can be effectively

---

2For graphs with multilabeled nodes, clustered anomalies and adversarial attacks are not defined and hence, these graphs are not included in the respective experiments.
employed as a preprocessing step to flag adversarial input to a graph neural network.

**Clustered anomalies**

Next, we consider identifying clusters of anomalous nodes Fig. 1.2b. Towards generating clustered anomalies, we select a cluster of connected nodes $\mathcal{A}$ using [29], and set all labels as $y_n = c, \forall n \in \mathcal{A}$, where $c$ is the least common label in $\mathcal{A}$. The anomalous nodes satisfy the homophily property inside the corrupted cluster that further challenges their detection.

Table 2.3 reports the AUC values of anomaly detection algorithms for identifying clustered anomalies. The performance of competing algorithms is heavily affected since within the affected cluster the anomalous nodes appear as nominal. On the other hand, our novel approach utilizes random sampling and consensus strategies, and markedly outperforms the baseline schemes in identifying the ground-truth anomalies.

**Random walk-based anomalies**

We test the algorithms in identifying homophilic random-walk based anomalies. To generate these we select a subset of $|\mathcal{A}|$ nodes at random, and alter their labels. For each $n \in \mathcal{A}$, we perform a random walk of length $k = 10$, and replace $y_n$ with the label of the landing node. Hence, we modify the labels of the targeted nodes in $\mathcal{A}$ as prescribed by the random walk model. The resulting nodes violate the homophily property.

Fig. 2.2 plots the AUC values of various methods with increasing $K$ on 6 benchmark graphs. Evidently, GraphSAC outperforms alternatives while the performance of all methods degrades slightly as $K$ increases. Furthermore, Fig. 2.3 reports the runtime of the best performing algorithms for $K = 20$ in Fig. 2.2 across all graphs. Evidently, GraphSAC is significantly faster than these competing methods.

**Parameter sensitivity.** Sensitivity of GraphSAC to $R, \tau$ is reported in Fig. 2.4. GraphSAC’s AUC performance is stable around the preselected values. As deduced in the discussion after Theorem 3, the number of iterations $R$ for a reliable anomaly score ranking is relatively small. A large enough ratio of correctly classified nodes $\tau$, aides GraphSAC in discarding contaminated samples since these will result in a large number of mis-classified nodes. As expected for small values of $\tau$ all the $R$ samples are accepted and the AUC performance decreases.

**On the practical interpretation of Theorem 2.** Fig. 2.5a reports the AUC performance of
Figure 2.4: GraphSAC’s sensitivity to $R, \tau$ for the Citeseer dataset with $K = 80$ in Fig. 2.2.

Figure 2.5: X-axis denotes the fraction of samples $S/N$ and y-axis represents the fraction of anomalies $K/N$.

GraphSAC for Citeseer in Fig. 2.2 with varying number of anomalies $K$ and sampling size $S$, where darker boxes indicate larger values of AUC. Observe that the AUC increases for larger $S$. For larger $K$ the AUC drops as expected, but this can be rectified by increasing $S$. Fig. 2.5b demonstrates the maximum number of anomalies $K_m := \max_r K^{(r)}$ that existed in the contaminated samples that were misclassified as nominal $b^{(r)} = 1$, where $K^{(r)} := |L^{(r)} \cup A|$. Notice $K_m$ in Fig. 2.5b is significantly smaller than the total number of anomalies that is $K = 17, 33, 83$, respectively. Hence, even though there are contaminated samples, the number anomalies in $L$ is so small that does not affect the AUC values in Fig. 2.5a This result is reflected on the assumptions of Theorem 2.
Figure 2.6: Classification accuracy in Cora with $|\mathcal{M}| = 100$

Figure 2.7: Classification accuracy in Citeseer under anomalies.
Figure 2.8: Classification accuracy in Polblogs under anomalies.
2.5.2 Robust semi-supervised learning

This section tests the performance gains of using GraphSAC to robustify SSL. The performance of the approaches is evaluated in terms of classification accuracy over the nodes in the unlabeled set \( U := \mathcal{V} - \mathcal{M} \). To account for the random sampling in the sampled sets, the reported results are averaged over 100 Monte-Carlo iterations.

For the label prediction step \( g(\cdot, \cdot) \) in (2.21), three state-of-the-art SSL algorithms are considered, namely i) the personalized Pagerank \[46\]; and the deep learning approaches ii) the graph convolutional network (GCN) \[80\] and iii) the graph attention network (GAT) \[127\]. Since, we are interested in comparing the original SSL algorithms with their GraphSAC-SSL versions, the same parameters are used for SSL and GraphSAC-SSL (number of layers, neurons).

Before studying the effect of anomalies in SSL, the performance of GraphSAC-SSL is evaluated on the original data, without perturbing the data. For this experiment, the 'Cora' dataset is considered and the sample set \( \mathcal{M} \) is selected at random with \(|\mathcal{M}| = 100\). Fig. 2.6 plots the accuracy of the original SSL and their GraphSAC-SSL counterparts, over the size of the 'clean' label set \( \mathcal{M}_c \); see (2.21). Note that the original SSL methods utilize all the sampled labels \( \mathcal{M} \), whereas GraphSAC-SSL utilizes only the filtered \( \mathcal{M}_c \). Although GraphSAC-SSL methods use less training labels, their accuracy is slightly better than the original methods that use the full training set. This finding suggests that some labels in \( \mathcal{M} \) may hurt the prediction performance, which are judiciously removed by GraphSAC to enhance prediction performance.

Next, we examine the performance gains of GraphSAC-SSL, when random walk-based anomalies are included in \( \mathcal{M} \); see also Sec. 2.5.1. For this experiment the following datasets are considered Citeseer, Polblogs, Cora, Pubmed. Figs. 2.7-2.10 showcase the accuracy of GraphSAC-SSL and SSL methods for different number of anomalies \( K \), and sizes of sampling sets \(|\mathcal{M}|\). On the x-axis, GraphSAC-SSL is evaluated for different sizes of retained sampling sets \( \mathcal{M}_c \). It is observed, that GraphSAC-SSL outperforms the corresponding SSL approaches for a large range of \( \mathcal{M}_c \). Particularly, Fig. 2.8 (c) shows that GraphSAC enhances the classification performance of the base SSL approaches by 30% from 0.7 to 0.9, while using only 16% of the sampled nodes in \( \mathcal{M} \), while 50% of the sampled nodes are perturbed. These results testify that GraphSAC endows SSL with increased robustness, especially in the presence of perturbed labels.

---

3Since the datasets do not contain features, for the deep learning approaches an identity matrix is used.
4Since the original SSL approaches use all sampled nodes \( \mathcal{M} \) for training, their accuracy performance does not change over \(|\mathcal{M}_c|\) in Figs. 2.6, 2.10.
5The deep learning approaches are not appropriate for graphs with multilabeled nodes as
Figure 2.9: Classification accuracy in Cora under anomalies.
Figure 2.10: Classification accuracy in Pubmed under anomalies.
in the node set.

2.6 Conclusion

The present chapter introduced graph-based random sampling and consensus approach to effectively detect anomalous nodes in graphs. Performance of the proposed scheme was evaluated both analytically and through simulated tests on multiple real datasets.
Chapter 3

Tensor-graph convolutional networks

A task of major importance in the interplay between machine learning and network science is semi-supervised learning (SSL) over graphs. This chapter develops a deep learning framework for SSL over a collection of graphs with applications to both multi-relational data and robust learning.

3.1 Related work

While significant progress has been achieved for this problem, most works consider that the relation among the nodal variables is represented by a single graph. This may be inadequate in many contemporary applications, where nodes may engage in multiple types of relations \[81\], motivating the generalization of traditional SSL approaches for single-relational graphs to multi-relational graphs\[^1\]. In the particular case of protein interaction networks, each layer of the graph could correspond to a different type of tissue (brain, muscle...). Alternatively, in a social network, each layer of the graph would capture a specific form of social interaction, such as friendship, family bonds, or coworker-ties \[129\]. Such graphs can be represented in a tensor graph, where each slab of the tensor corresponds to a single relation. Albeit their ubiquitous presence, development of SSL methods that account for multi-relational networks is only in its infancy, see, e.g., \[67\][81]. This work develops a novel robust deep learning framework for SSL over multi-relational graphs.

Graph-based SSL methods typically assume that the true labels are “smooth” with respect

\[^1\]Many works in the literature refer to these graphs as multi-layer graphs \[81\].
to the underlying network structure, which naturally motivates leveraging the topology of the network to propagate the labels and increase classification performance. Graph-induced smoothness may be captured by kernels on graphs \[12, 59\]; Gaussian random fields \[146\]; or low-rank parametric models based on the eigenvectors of the graph Laplacian or adjacency matrices \[20, 24, 33, 95, 119\]. Alternative approaches use the graph to embed the nodes in a vector space, and classify the points \([13, 131]\). More recently, another line of works postulates that the mapping between the input data and the labels is given by a neural network (NN) architecture that incorporates the graph structure and generalizes the typical convolution operations; see e.g., \([17, 41, 60, 76, 77, 80, 115, 122]\). The parameters describing the graph convolutional NN (GCN) are then learned using labeled examples and feature vectors, and those parameters are finally used to predict the labels of the unobserved nodes. See, e.g., \([17, 80, 127, 137]\), for state-of-the-art results in SSL when nodes are accompanied with additional features.

With the success of GCNs on graph learning tasks granted, recent results indicate that perturbations of the graph topology can severely deteriorate their classification performance \([30, 136, 150]\). Such uncertainty in the graph topology may be attributed to several reasons. First, oftentimes the graph is implicit and data-driven methods are employed for learning the topology \([45]\). However, each method relies on a different model and assumptions, and in the absence of a ground truth graph selecting the appropriate graph-learning technique is challenging. An inappropriate model may introduce model-based perturbations to the learned graph. Moreover, consider the case of random graph models \([98]\), where the graph is a particular realization of the model and edges may be randomly perturbed. Similarly, this idea is also relevant in adversarial settings, where the links of the nominal graph are corrupted by some foe that aims to poison the learning framework. Adversarial perturbations target a subset of nodes and modify their links to promote mis-classification of targeted nodes \([133]\). The designed graph perturbations are “unnoticeable”, which is feasible so long as the degree distribution of the perturbed graphs is similar to the initial distribution \([150]\). GCNs learn nodal representations by extracting information within local neighborhoods. These learned features may be significantly perturbed if the neighborhood is altered. Hence, this vulnerability of GCNs challenges their deployment in critical applications dealing with security or healthcare, where robust learning is of major importance. Defending against adversarial, random, or model-based perturbations may unleash the potential of GCNs and broaden the scope of machine learning applications altogether.
3.1.1 Contributions

The contribution of this work is five-fold

C1. A tensor-GCN architecture is developed that accounts for multi-relational graphs. Learnable coefficients are introduced allowing the flexible model to adapt to the multiple graphs and identify the underlying structure of the data.

C2. A multi-hop convolution together with a residual feed of the data for each of the graphs are proposed, broadening the class of (graph signal) transformations the GCN implements and, hence, facilitating the diffusion of the features across the graph. In the training phase suitable (graph-based) regularizers are considered to avoid overfitting and further capitalize on the graph topology.

C3. For datasets where nodes are involved in different relations and (multi-relational) data associated with the different graphs exist, the proposed TGCN architecture provides a powerful learning framework to carry out predictions that leverage the information codified in the multiple graphs.

C4. Our TGCN also facilitates robust SSL for single or multi-relational data when the underlying topology is perturbed. Model-based, random, and adversarial perturbations are considered and our TGCN is adapted to robustify SSL over these perturbed graphs. To defend against adversaries a novel edge-dithering (ED) approach is developed that generates ED graphs by sampling edges of the original graph with probabilities selected to enhance robustness.

C5. Numerical tests with multi-relational protein networks showcase the merits of the proposed tensor-graph framework. Further experiments under noisy features, noisy edge weights, and random as well as adversarial edge perturbations verify the robustness properties of our novel approach.

3.2 SSL over multi-relational graphs

Consider a network of $N$ nodes, with nodal (vertex) set $V := \{v_1, \ldots, v_N\}$, connected through $I$ relations. The connectivity at the $i$th relation is captured by the $N \times N$ matrix $S_i$, and the
scalar $S_{nn'i}$ represents the influence of $v_n$ to $v_{n'}$ under the $i$th relation. The matrices $\{S_i\}_{i=1}^I$ are collected in the $N \times N \times I$ tensor $S$. To complement the examples already provided in the introduction, and focusing on the case social networks, each $i$ could for instance represent a relation via a particular online social app such as Facebook, LinkedIn, and Twitter; see Fig.1.3. Regardless of the particular application, the graph-induced neighborhood of $v_n$ for the $i$th relation is

$$\mathcal{N}_n^{(i)} := \{n' : S_{nn'i} \neq 0, \; v_{n'} \in \mathcal{V}\}.$$ (3.1)

We associate an $F \times 1$ feature vector $x_n$ with the $n$th node, and collect those vectors in the $N \times F$ feature matrix $X := [x_1^\top, \ldots, x_N^\top]^\top$. The entry $X_{np}$ may denote, for example, the salary of the $n$th individual in the LinkedIn social network.

We also consider that each node $n$ has a label of interest $y_n \in \{0, \ldots, K - 1\}$, which, in the last example, could represent the education level of a person. In SSL we have access to the labels only at a subset of nodes $\{y_n\}_{n \in \mathcal{M}}$, with $\mathcal{M} \subset \mathcal{V}$. This partial availability may be attributed to privacy concerns (medical data); energy considerations (sensor networks); or unrated items (recommender systems). The $N \times K$ matrix $Y$ is the “one-hot” representation of the true nodal labels, that is, if $y_n = k$ then $Y_{n,k} = 1$ and $Y_{n,k'} = 0, \forall k' \neq k$.

The goal of this chapter is to develop a robust tensor-based deep learning architecture for SSL over multi-relational graphs. Given $X$, the proposed network maps each node $n$ to a corresponding label $y_n$ and, hence, estimates the unavailable labels.

### 3.3 Proposed TGCN architecture

Deep learning architectures typically process the input information using a succession of $L$ hidden layers. Each of the layers is composed of a conveniently parametrized linear transformation, a scalar nonlinear transformation, and, oftentimes, a dimensionality reduction (pooling) operator. The intuition is to combine nonlinearly local features to progressively extract useful information [47]. GCNs tailor these operations to the graph that supports the data [17], including the linear [31], nonlinear [31], and pooling [41] operators. In this section, we describe the architecture of our novel multi-relational TGCN, which inputs the known features at the first layer and outputs the predicted labels at the last layer. We first present the operation of the
Let us consider an intermediate layer (say the $l$th one) of our architecture. The output of that layer is the $N \times I \times P(l)$ tensor $\tilde{Z}^{(l)}$ that holds the $P(l) \times 1$ feature vectors $\tilde{z}_{ni}^{(l)}$, $\forall n, i$, with $P(l)$ being the number of output features at $l$. Similarly, the $N \times I \times P(l-1)$ tensor $\tilde{Z}^{(l-1)}$ represents the input to the layer. Since our focus is on predicting labels on all the nodes, we do not consider a dimensionality reduction (pooling) operator in the intermediate layers. The mapping from $\tilde{Z}^{(l-1)}$ to $\tilde{Z}^{(l)}$ can then be split into two steps. First, we define a linear transformation that maps the $N \times I \times P(l)$ tensor $\tilde{Z}^{(l-1)}$ into the $N \times I \times P(l)$ tensor $Z^{(l)}$. A scalar nonlinear transformation $\sigma(\cdot)$ is applied to $Z^{(l)}$ as follows

\[
\tilde{Z}_{\text{imp}}^{(l)} := \sigma(\tilde{Z}^{(l)})...
\]

Collecting all the elements in (3.2), we obtain the output of the $l$th layer $\tilde{Z}^{(l)}$. A common choice for $\sigma(\cdot)$ is the rectified linear unit (ReLU), i.e. $\sigma(c) = \max(0, c)$ [47].

Hence, the main task is to define a linear transformation that maps $\tilde{Z}^{(l-1)}$ to $Z^{(l)}$ and is tailored to our problem setup. Traditional convolutional NNs (CNNs) typically consider a small number of trainable weights and then generate the linear output as a convolution of the input with these weights [47]. The convolution combines values of close-by inputs (consecutive time instants, or neighboring pixels) and thus extracts information of local neighborhoods. Permeating the benefits of CNNs to the graph domain, GCNs replace the convolution with a graph filter whose parameters are also learned [17]. This preserves locality, reduces the degrees of freedom of the transformation, and leverages the structure of the graph.

In the following three subsections we present the structure of the novel tensor-graph linear transformation and discuss in detail how the multi-relational graph is taken into account.

**Neighborhood aggregation module (NAM).** First, we consider a neighborhood aggregation module that, for each of the graphs, combines linearly the information available locally within each graph neighborhood. Since the neighborhood depends on the particular relation (3.1), we
obtain for the $i$th relation and $n$th node

$$h_{ni}^{(l)} := \sum_{n' \in \mathcal{N}_h^{(ii)}} S_{nn'} z_{n'i}^{(l-1)}.$$ (3.3)

While the entries of $h_{ni}^{(l)}$ depend only on the one-hop neighbors of $n$ (one-hop diffusion), the successive application of this operation across layers will increase the reach of the diffusion, spreading the information across the network. Specifically, consider the $r$th power of the matrix $S^r$. Indeed, the vector $S^r x$ holds the linear combination of the values of $x$ in the $r$-hop neighborhood \cite{95}. After defining the matrices $S_i^{(r)} := S^r_i$ for $r = 1, \ldots, R$ and $i = 1, \ldots, I$, consider the following parametrized mapping

$$h_{ni}^{(l)} := \sum_{r=1}^{R} \sum_{n' = 1}^{N} C_i^{(r,l)} S_{nn'}^{(r)} z_{n'i}^{(l-1)},$$ (3.4)

where the learnable coefficients $C_i^{(r,l)}$ weight the effect of the corresponding $r$th hop neighbors of node $n$ according to relation $i$. At the $l$th layer, the coefficients $\{C_i^{(r,l)}\}_{r,l}$ are collected in the $R \times I$ matrix $C^{(l)}$. The proposed transformation in (3.4) aggregates the diffused signal in the $R$-hop neighborhoods per $i$; see also Fig. 3.1.

**Graph adaptive module (GAM).** The extracted feature $h_{ni}^{(l)}$ captures the diffused input per relation $i$. The importance of a particular feature or relation will depend on the inference task at hand. For example, in predicting the voting preference the friendship network may be more important than the coworker relation; cf. Fig. 1.3 As a result, the learning algorithm should be able to adapt to the prevalent features. To that end, we adapt to the different relations and combine $h_{ni}^{(l)}$ across $i$ as follows

$$g_{ni}^{(l)} := \sum_{i' \in I} R_{ii'n}^{(l)} h_{ni'}^{(l)},$$ (3.5)

where $R_{ii'n}^{(l)}$ mixes the outputs at different graphs. Another key contribution of this work is the consideration of the graph-mixing weights $\{R_{ii'n}^{(l)}\}_{i,i',n}$, which can be collected in the $I \times I \times N$ tensor $R^{(l)}$, as a training parameter. The graph-mixing weights endow our TGCN with the ability of learning how to combine and adapt to the different relations encoded in the multi-relational graph; see also Fig. 3.2. Clearly, if prior information on the dependence among
Neighborhood aggregation module

\[ h_{n1}^{(0)} \]

\[ h_{n2}^{(0)} \]

\[ h_{n3}^{(0)} \]

Figure 3.1: The NAM combines the features using the multi-relational graph. The picture focuses on node \( n \) and illustrates the case where \( R \)-hop neighbors (with \( R = 2 \)) are considered. Note that, as shown in the picture, the local neighborhood is not the same across the different graphs.

relations exists, this can be used to constrain the structure \( R^{(l)} \) (e.g., by imposing to be diagonal or sparse). The graph-adaptive combination in (3.5) allows for different \( R_{i'j'n} \) per \( n \). Considering the same \( R \) for each \( n \), that is \( R_{i'j'n}^{(l)} = R_{i'j'}^{(l)} \), results in a design with less parameters at the expense of reduced flexibility. For example, certain voters may be affected more significantly from their friends whereas others from their coworkers. Using the adaptive module our network can achieve personalized predictions.

Feature aggregation module (FAM). Next, the extracted graph adaptive diffused features are

Figure 3.2: The GAM combines the features per \( i \), based on the trainable coefficients \( \{R_{i'i'n}\} \).
mixed using learnable scalars $W_{nipp'}$ as follows

$$Z_n^{(l)} := \sum_{p'=1}^{P(l-1)} W_{nipp'} G_{nipp'}^{(l)}, \quad (3.6)$$

for all $(n, i, p)$ and where $G_{nipp'}^{(l)}$ represents the $p'$th entry of $g_n^{(l)}$. The $N \times I \times P^{(l)} \times P^{(l-1)}$ tensor $W^{(l)}$ collects the feature mixing weights $\{W_{nipp'}^{(l)}\}_{(n,i,p,p')}$. The linear transformations that transform the input tensor $Z_{n}^{(l-1)}$ to $Z^{(l)}$ are summarized as follows

$$Z^{(l)} := f(\hat{Z}^{(l-1)}; \theta_{z}^{(l)}), \quad \text{with}$$

$$\theta_{z}^{(l)} := [\text{vec}(W^{(l)}); \text{vec}(R^{(l)}); \text{vec}(C^{(l)})]^\top, \quad (3.8)$$

where $f$ has been used to denote the successive application of the three linear modules just introduced (namely NAM, GAM and FAM) and $\theta_{z}^{(l)}$ collects the learnable weights involved in those modules [cf. (3.4)-(3.6)].

### 3.3.2 Residual GCN layer

Successive application of $L$ TGCN layers diffuses the input $X$ across the $LR$-hop graph neighborhood, cf. (3.3). However, the exact size of the relevant neighborhood is not always known a priori. To endow our architecture with increased flexibility, we propose a residual TGCN layer that inputs $X$ at each $l$ and, thus, captures multiple types of diffusion\footnote{This is also known as a skip connection \cite{53}}. Hence, the linear operation in (3.7) is replaced by the residual (auto-regressive) linear tensor mapping \cite[Ch. 10]{47}

$$Z^{(l)} := f(\hat{Z}^{(l-1)}; \theta_{z}^{(l)}) + f(X; \theta_{x}^{(l)}) \quad (3.9)$$

where $\theta_{x}^{(l)}$ encodes trainable parameters, cf. (3.8). When viewed as a transformation from $X$ to $Z^{(l)}$, the operator in (3.9) implements a broader class of graph diffusions than the one in (3.7). If, for example, $l = 3$ and $k = 1$, then the first summand in (3.9) is a 1-hop diffusion of a signal that corresponded to a 2-hop (nonlinear) diffused version of $X$ while the second summand diffuses $X$ in one hop. At a more intuitive level, the presence of the second summand
also guarantees that the impact of \( \mathbf{X} \) in the output does not vanish as the number of layers grow. The auto-regressive mapping in (3.9) facilitates the application of our architecture in scenarios with time-varying inputs and labels. Specifically, with \( t \) denoting the time index and given time-varying data \( \{ \mathbf{X}_t \}_T^t \), one would set \( l = t \), replace \( \mathbf{X} \) in (3.9) with \( \mathbf{X}^{(t)} \) and then set \( \mathbf{X}^{(t)} = \mathbf{X}_t \). This will be studied in detail in our future work towards predicting dynamic processes over multi-relational graphs.

### 3.3.3 Initial and final layers

Regarding layer \( l = 1 \), the input \( \tilde{\mathbf{Z}}^{(0)} \) is defined as

\[
\tilde{z}^{(0)}_{ni} = x_n \quad \text{for all} \quad (n, i).
\]  

(3.10)

On the other hand, the output of our graph architecture is obtained by taking the output of the layer \( l = L \) and applying

\[
\hat{\mathbf{Y}} := g(\tilde{\mathbf{Z}}^{(L)}; \theta_g),
\]  

(3.11)

where \( g(\cdot) \) is a nonlinear function, \( \hat{\mathbf{Y}} \) is an \( N \times K \) matrix, \( \hat{Y}_{n,k} \) represents the probability that \( y_n = k \), and \( \theta_g \) are trainable parameters. The function \( g(\cdot) \) depends on the specific application, with the normalized exponential function (softmax) being a popular choice for classification problems that is

\[
\hat{Y}_{n,k} = \frac{\exp \tilde{z}_{ni,k}^{(L)}}{\sum_{k=1}^{K} \exp \tilde{z}_{ni,k}^{(L)}}.
\]  

(3.12)

For notational convenience, the global mapping \( \mathcal{F} \) from \( \mathbf{X} \) to \( \hat{\mathbf{Y}} \) dictated by our TGCN architecture is denoted as

\[
\hat{\mathbf{Y}} := \mathcal{F}(\mathbf{X}; \{ \theta_z^{(l)} \}_{l=1}^{L}, \{ \theta_x^{(l)} \}_{l=1}^{L}, \theta_g),
\]  

(3.13)

and represented in the block diagram depicted in Fig. 3.3.
Figure 3.3: TGCN with \( L \) hidden (black) and one output (red) layers. The input \( X \) contains a collection of features per node and the output to be predicted is the probability of each node to belong to each of the \( K \) classes (labels) considered. Each layer of the TGCN is composed of our three novel modules (NAM, GAM, FAM) described in equations (3.4), (3.5), and (3.6). Notice the skip connections that input \( X \) to each layer [cf. (3.9)].

### 3.3.4 Training and graph-smooth regularizers

The proposed architecture depends on the weights in (3.9) and (3.11). We estimate these weights by minimizing the discrepancy between the estimated labels and the given ones. Hence, we arrive at the following minimization objective

\[
\min \left\{ \theta(z)_L^{(l)}, \theta(x)_L^{(l)}, \theta_g \right\} \mathcal{L}_{tr}(\hat{Y}, Y) + \mu_1 \sum_{i=1}^{L} \text{Tr}(\hat{Y}^\top S_i \hat{Y}) + \\
+ \mu_2 \rho(\{\theta_z^{(l)}\}_{l=1}^{L}, \{\theta_x^{(l)}\}_{l=1}^{L}) + \lambda \sum_{l=1}^{L} \|R^{(l)}\|_1
\]

s.t. \( \hat{Y} = \mathcal{F}(X; \{\theta_z^{(l)}\}_{l=1}^{L}, \{\theta_x^{(l)}\}_{l=1}^{L}, \theta_g) \).

In our classification setup, a sensible choice for the fitting cost is the cross-entropy loss function over the labeled examples that is \( \mathcal{L}_{tr}(\hat{Y}, Y) := -\sum_{n \in M} \sum_{k=1}^{K} Y_{nk} \ln \hat{Y}_{nk} \).

The first (graph-based) regularizer in (3.14) promotes smooth label estimates over the graphs [123], and \( \rho(\cdot) \) is an \( \ell_2 \) norm over the TGCN parameters typically used to avoid overfitting [47]. Finally, the \( \ell_1 \) norm in the third regularizer encourages learning sparse mixing coefficients, and hence it promotes activating only a subset of relations per \( l \). The learning algorithm will
assign larger combining weights to topologies that are most appropriate for the given data. A backpropagation algorithm \cite{112} is employed to minimize (3.14). The computational complexity of evaluating (3.9) scales linearly with the number of nonzero entries in $S$ (edges) [cf. (3.3)].

To recap, while most of the works in the GCN literature use a single graph with one type of diffusion \cite{17, 80}, in this section we have proposed a (residual) TGCN architecture that: i) accounts for a collection of graphs defined over the same set of nodes; ii) diffuses the signals across each of the different graphs; iii) combines the signals at the different graphs using adaptive (learnable) coefficients; iv) implements the simple but versatile residual tensor mapping (3.9); and v) includes several types of graph-based regularizers.

### 3.4 Robust GCNs via tensor-graphs

In the previous section, the nodes were involved in $I$ different types of relations, with each slab of our tensor graph $S$ representing one of those relations. In this section, the proposed tensor-graph architecture is applied to robustify classical single-graph GCNs. Consider that the nodes are involved in a single relation represented by the graph $\tilde{G}$ and $\tilde{G}$ does not necessarily represent the true graph but an approximate (nominal) version of it. That can be the case, for example, in applications involving random graph models \cite{15, 52, 98}, where $\tilde{G}$ is a particular realization but other realizations could be considered as well. Similarly, this idea is also relevant in adversarial settings, where the links of the nominal graph $\tilde{G}$ are corrupted by some foe (see Fig. 3.4 for an illustration of this setup along with additional details). Our approach in this section is to use $\tilde{G} = (V, A)$ to generate a set of $I$ candidate graphs $\{\tilde{G}_i\}_{i=1}^I = (V, A_i)$ and then collect the adjacency matrices of those graphs in the tensor $S$. Clearly, this approach can also be used for multi-relational graphs, generating multiple candidate graphs for each relation. The next subsections elaborate on three scenarios of particular interest.

#### 3.4.1 Robustness to the graph learning method

While in applications dealing with communications, power or transportation systems the network connecting the different nodes may be explicitly known, in a number of scenarios the graph is implicit and must be learned from observed data. Several methods to infer the topology exist, each relying on a different model that relates the graph with the properties of the data \cite{45}. Since in most applications a ground-truth graph does not exist, the issue of how to select the
appropriate graph-learning method arises. More importantly for the setup considered in this chapter, the particular selected method (and, hence, the particular graph) will have an impact on the performance of the GCN.

Consider first the $\kappa$-nearest neighbors ($\kappa$-NN) method, which is employed to construct graphs in various applications, including collaborative filtering, similarity search, and many others in data mining and machine learning [35]. This method typically computes the link between $n$ and $n'$ based on a distance between their nodal features. For instance, for the Euclidean distance we simply have $d(n, n') = \| x_n - x_{n'} \|_2^2$. Then, for each node $n$ the distances with respect to all other nodes $n' \neq n$ are ranked and $n$ is connected with the $\kappa$ nodes with the smallest distances $\{d(n, n')\}$. However, selecting the appropriate $\kappa$ and distance metric $d(\cdot, \cdot)$ is often arbitrary and may not generalize well to unseen data, especially if the learning system operates in an online fashion. Hence, our approach to robustify SSL in that scenario is to consider a tensor graph where each slab corresponds to a graph constructed using a different value of $\kappa$ and (or) distance.

A similar challenge arises in the so-called correlation network methods [45]. In this case, the graph is learned based on the correlation between the data observed at each pair of nodes. Among other things, this requires comparing the observed sample correlation to a threshold $\eta$ and, then, declare that the edge exists if the measured correlation is above $\eta$. Selecting the proper value for $\eta$ is oftentimes arbitrary and can compromise the prediction performance of the GCN. Similarly, there are applications, including those related to Markov random fields, where correlation networks are not appropriate but partial correlation networks (which look at the inverse of the covariance matrix [45]) are.

In such cases, we can collect the multiple learned graphs, originating from possibly different methods, as slabs of $S$, and then train our TGCN architecture. Depending on the application at hand, it may be prudent to include in the training a block-sparsity penalty on the coefficients $\mathbf{R}$, so that we exploit available prior on the most appropriate graphs.

### 3.4.2 Robustness to edge attacks via edge dithering

The ever-expanding interconnection of social, email, and media service platforms presents an opportunity for adversaries manipulating networked data to launch malicious attacks [2, 48, 150]. Perturbed edges modify the graph neighborhoods, which leads to significant degradation of the performance achieved by GCNs. In the voting network (see Fig. [1.3]), some of the edges may be adversarially manipulated so that the voters are influenced in a specific direction. This section
explains how to use our TGCN to deal with learning applications for which the graph has been adversarially perturbed.

In particular, we consider an edge-dithering (ED) module that, given the nominal graph, creates a new graph by randomly adding/removing links with the aim to restore a node’s initial graph neighborhood. Dithering in visual and audio applications, refers to the intentional injection of noise so that the quantization error is converted to random noise, which can be handled more easily [126]. Therefore, the approach that we advocate is to use an instance of our TGCN architecture where each of the slabs of the tensor $S$ corresponds to a graph that has been obtained after dithering some of the links of the nominal (potentially compromised) graph $\tilde{G}$.

Mathematically, given the (perturbed) graph $\tilde{G} = (V, \tilde{A})$, we generate $I$ ED graphs $\{G_i\}_{i=1}^I$, with $G_i = (V, A_i)$ and where the edges of the auxiliary graph $A_i$ are selected in a probabilistic fashion as follows

$$A_{n,n';i} = \begin{cases} 1 \text{ wp. } & q_1 \delta(\tilde{A}_{n,n'}=1) (1 - q_2) \delta(\tilde{A}_{n,n'}=0) \\ 0 \text{ wp. } & q_2 \delta(\tilde{A}_{n,n'}=0) (1 - q_1) \delta(\tilde{A}_{n,n'}=1). \end{cases}$$

In the previous expression, $\delta(\cdot)$ is the indicator function and the dithering probabilities are set as
If \( n \) and \( n' \) are connected in \( \bar{G} \), the edge connecting \( n \) with \( n' \) is deleted with probability \( 1 - q_1 \). Otherwise, if \( n \) and \( n' \) are not connected in \( \bar{G} \) i.e. \( \bar{A}_{n,n'} = 0 \), an edge between \( n \) and \( n' \) is inserted with probability \( 1 - q_2 \). The ED graphs give rise to different neighborhoods \( N_n^{(i)} \), and the role of the ED module is to ensure that the unperturbed neighborhood of each node will be present with high probability in at least one of the \( I \) graphs. For clarity, we formalize this intuition in the following remark.

**Remark 1.** With high probability, there exists \( \bar{G}_i \) such that a perturbed edge will be restored to its initial value. This means that there exists an ED graph \( i \) such that \( A_{n,n',i} = A_{n,n'} \). Since, each \( \bar{G}_i \) is independently drawn, it holds that

\[
\Pr(\prod_{i=1}^I \delta(A_{n,n',i} = 1) | \bar{A}_{n,n'} = 1, A_{n,n'} = 0) = q_1^I
\]
\[
\Pr(\prod_{i=1}^I \delta(A_{n,n',i} = 0) | \bar{A}_{n,n'} = 0, A_{n,n'} = 1) = q_2^I.
\]

That is, as \( I \) increases, the probability that none of the graphs gets the true value for the perturbed link decreases exponentially. By following a similar argument, one can argue that, as \( I \) increases, the probability that none of the graphs recovers the original neighborhood structure decreases, so that there would exist an ED graph \( i \) such that \( N_n^{(i)} = N_n \). More importantly, since our architecture linearly combines (outputs of) different graphs, this will effectively span the range of graphs that we are able to represent, rendering the overall processing scheme less sensitive to adversarial edge perturbations. Indeed, numerical experiments with adversarial attacks will demonstrate that, even with a small \( I \), the use of ED significantly boosts classification performance. The operation of the ED module is illustrated in Fig. 3.4.

### 3.4.3 Learning over random graphs

Uncertainty is ubiquitous in nature and graphs are no exception. Testament to this fact are the efforts to develop meaningful and tractable models for random graphs stemming not only from the graph-theory community (from the early Erdős-Rényi models to more-recent low-rank graphon generalizations [15]), but also from the network-science (e.g., preferential attachment models [98, Ch. 12-16]) and statistics (e.g., exponential random graph models [52]) communities. Those random graph models provide an excellent tool for studying structural features of networks, such as giant and small components, degree distributions, path lengths, and so forth. Equally
important, they provide parsimonious parametric models that can be leveraged for inference and inverse problems. That is the case in scenarios where we have access to limited graph-related observations such as the induced graph at a subset of nodes, or the mean and variance of some graph motifs. In those cases, inferring the full graph can be infeasible, but one can postulate a particular random graph model and use the available observations to infer the parameters that best fit the data.

A natural issue is, then, how to use such random graph models for the purpose of learning from an (incomplete) set of graph signals in the context of GCNs. A number of alternatives exist, including, for example, implementing a multi-layer graph convolutional architecture where, at each layer, a different realization of the graph is used. A different approach, which is the one advocated here, is to leverage the TGCN architecture put forth in this work. In this case, the idea is to draw \( I \) realizations of the random graph model, collect those in the \( N \times N \times I \) tensor \( S \), and train a TGCN. This way, we guarantee that each layer considers not one, but multiple realizations of the graph. Clearly, if we consider an online setup where the layers of the GCN can be associated with time, the proposed model can be related with importance sampling and particle filtering approaches, with each slab of the tensor \( S \) representing a different particle of the graph probability space \([19]\). This hints at the possibility of developing TGCN schemes for the purpose of nonlinear Bayesian estimation over graphs. While certainly of interest, we leave that as future work.

### 3.5 Numerical tests

This section tests the performance of TGCN in learning from multiple potentially perturbed graphs and provides tangible answers to the following research questions.

**Q1.** How does TGCN compare to state-of-the-art methods for SSL over multi-relational graphs?

**Q2.** How can TGCN leverage topologies learned from multiple graph-learning methods?

**Q3.** How robust is TGCN compared to GCN under noisy features, noisy edge weights, and random as well as adversarial edge perturbations?

**Q4.** How sensitive is TGCN to the parameters of the ED module (i.e., \( q_1, q_2 \) and \( I \))?
Figure 3.5: Classification accuracy on the synthetic (a)-(b) and ionosphere (c)-(d) graphs described in Sec. 3.5.1 as the noise level in the features [cf. (3.17)] or in the links [(3.16)] varies. Panels (a) and (c) show the classification accuracy for noisy features while panels (b) and (d) show the same metric as the power of the noise added to the graph links varies.

To that end, and unless it is otherwise stated, we test the proposed TGCN with $R = 2$, $L = 3$, $P^{(1)} = 64$, $P^{(2)} = 8$, and $P^{(3)} = K$. The regularization parameters $\{\mu_1, \mu_2, \lambda\}$ are chosen based on the performance of the TGCN in the validation set for each experiment. For the training stage, an ADAM optimizer with learning rate 0.005 was employed [79], for 300 epochs with early stopping at 60 epochs. The simulations were run using TensorFlow [1] and the code is available online[5].

---

[3] An epoch is a cycle through all the training examples

[4] Training stops if the validation loss does not decrease for 60 epochs

[5] https://sites.google.com/site/vasioanidispw/github
3.5.1 SSL using multiple learned graphs

This section reports the performance of the proposed architecture when multiple learned graphs are employed and data are corrupted by noise. Oftentimes, the available topology and feature vectors might be noisy. In those cases, the observed $S$ and $X$ can be modeled as

$$S = S_{tr} + O_S \quad (3.16)$$
$$X = X_{tr} + O_X. \quad (3.17)$$

where $S_{tr}$ and $X_{tr}$ represent the true topology and features and $O_S$ and $O_X$ denote the corresponding additive perturbations. We draw $O_S$ and $O_X$ from a zero-mean uncorrelated multivariate Gaussian distribution with specified signal to noise ratio (SNR). The robustness of our method is tested in two datasets: i) A synthetic dataset of $N = 1000$ points that belong to $K = 2$ classes generated as $x_n \sim N(m_x, 0.4I)$ for $n = 1, \ldots, 1000$, with $F = 10$ and the mean vector $m_x \in \mathbb{R}^{F \times 1}$ being all zeros for the first class and all ones for the second one. ii) The ionosphere dataset, which contains $N = 351$ data points with $F = 34$ features that belong to $K = 2$ classes [32]. We generate $\kappa$-NN graphs by varying $\kappa$, and observe $|\mathcal{M}| = 200$ and $|\mathcal{M}| = 50$ nodes uniformly at random.

With this simulation setup, we test the different TGCNs in SSL for increasing SNR values (Figs. 3.5a, 3.5b, 3.5c, 3.5d). We deduce from the classification performance of our method in Fig. 3.5 that multiple graphs lead to learning more robust representations of the data, demonstrating the merits of the proposed tensor graph architecture.

3.5.2 Robustness of TGCNs to random graph perturbations

For this experiment, our TGCN utilizes our novel ED module and TGCN architecture to account for perturbations on the graph edges.

In this case, the experiments are run using three of the citation network datasets in [116]. The adjacency matrix of the citation graph is denoted as $S$, its nodes correspond to different documents from the same scientific category, and $S_{nn'} = 1$ implies that paper $n$ cites paper $n'$. Each document $n$ is associated with a label $y_n$, that indicates the document’s subcategory. “Cora” contains papers related to machine learning, “Citeseer” includes papers related to computer and information science, while “Pubmed” contains biomedical papers, see also Table 3.1. To
Table 3.1: List of citation graph datasets considered in Secs. 3.5.2 and 3.5.2 along with most relevant dimensions.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Nodes N</th>
<th>Classes K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>2,708</td>
<td>7</td>
</tr>
<tr>
<td>Citeseer</td>
<td>3,327</td>
<td>6</td>
</tr>
<tr>
<td>Pubmed</td>
<td>19,717</td>
<td>3</td>
</tr>
<tr>
<td>Polblogs</td>
<td>1,224</td>
<td>2</td>
</tr>
</tbody>
</table>

facilitate comparison, we reproduce the same experimental setup than in [80], i.e., the same split of the data in train, validation, and test sets. For this experiment, the perturbed graph \( \bar{A} \) is generated by inserting new edges in the original graphs between a random pair of nodes \( n, n' \) that are not connected in \( A \), i.e. \( A_{n,n'} = 0 \). This can represent, for example, documents that should have been cited but the authors missed. The added edges can be regarded as drawn from Bernoulli distribution. The TGCN utilizes the multiple graphs generated via the ED module with \( I = 10 \) samples, \( q_1 = 0.9 \), and \( q_2 = 1 \) since no edge is deleted in \( \bar{A} \).

Fig. 3.6 demonstrates the classification accuracy of the GCN [80] compared to that of the proposed TGCN as the number of perturbed edges is increasing. Clearly, our ED-TGCN is more robust than a classical GCN. Moreover, even when no edges are perturbed, the TGCN outperforms the GCN. This observation may be attributed to noisy links in the original graphs, which hinder classification performance. Furthermore, the SSL performance of the GCN significantly degrades as the number of perturbed edges increases, which suggests that GCN is challenged even by “random attacks”.

3.5.3 Robustness to adversarial attacks on edges

The original graphs in Cora, Citeseer, Pubmed, and Polblogs were perturbed using the adversarial setup in [150], where structural attacks are effected on attributed graphs. These attacks perturb connections adjacent to \( T \) a set of targeted nodes by adding or deleting edges [150]. Our ED module uses \( I = 10 \) sampled graphs with \( q_1 = 0.9 \), and \( q_2 = 0.999 \). For this experiment, 30% of the nodes are used for training, 30% for validation and 40% for testing. The nodes in \( T \) are in the testing set.

Table 3.2 reports the classification accuracy of the GCN and the proposed TGCN for different numbers of attacked nodes (\( |T| \)). Different from Fig. 3.6 where the classification accuracy over
Figure 3.6: Classification accuracy for the setup described in Sec. 3.5.2 as the number of perturbed edges increases.
Table 3.2: Classification accuracy for the setup described in Sec. 3.5.3 as the number of attacked nodes $|\mathcal{T}|$ increases.

| Dataset  | Method | Number of attacked nodes $|\mathcal{T}|$ |
|----------|--------|---------------------------------------|
|          |        | 20  30  40  50  60  70.99  56.00  61.49  56.39  58.99  58.66 |
| Citesee  | GCN    | 60.49  56.00  61.49  56.39  58.99  70.99  56.00  61.49  61.20  58.99 |
|          | TGCN   | 70.99  56.00  61.49  61.20  58.99  70.99  56.00  61.49  61.20  58.99 |
| Cora     | GCN    | 76.00  74.66  76.00  62.39  73.66  70.99  56.00  61.49  61.20  58.99 |
|          | TGCN   | 78.00  82.00  84.00  73.59  74.99  78.00  82.00  84.00  73.59  74.99 |
| Pubmed   | GCN    | 74.00  71.33  68.99  66.40  69.66  72.00  75.36  71.44  68.50  74.43 |
|          | TGCN   | 72.00  75.36  71.44  68.50  74.43  72.00  75.36  71.44  68.50  74.43 |
| Polblogs | GCN    | 85.03  86.00  84.99  78.79  86.91  84.00  88.00  91.99  78.79  86.91 |
|          | TGCN   | 84.00  88.00  91.99  78.79  92.00  84.00  88.00  91.99  78.79  92.00 |

Figure 3.7: SSL classification accuracy of the TGCN under varying edge creation prob. $q_1$, edge deletion prob. $q_2$, and number of samples $I$.

The test set is reported, Table 3.2 reports the classification accuracy over the set of attacked nodes $\mathcal{T}$. It is observed that the proposed TGCN is more robust relative to GCN under adversarial attacks [150]. This finding justifies the use of the novel ED in conjunction with the TGCN that judiciously selects extracted features originating from non-corrupted neighborhoods.

Fig. 3.7 showcases the sensitivity of the TGCN to varying parameters of the ED module for the experiment in Table 3.2 with the Cora and $|\mathcal{T}| = 30$. It is observed that the TGCN’s performance is relative smooth for certain ranges of the parameters. In accordance with Remark 2, notice that even for small $I$ the TGCN’s performance is increased significantly.
Table 3.3: List of protein-to-protein interaction datasets considered in Sec. 3.5.4 and their associated dimensions.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Nodes N</th>
<th>Features F</th>
<th>Relations I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic cells</td>
<td>4,487</td>
<td>502</td>
<td>144</td>
</tr>
<tr>
<td>Brain cells</td>
<td>2,702</td>
<td>81</td>
<td>9</td>
</tr>
<tr>
<td>Circulation cells</td>
<td>3,385</td>
<td>62</td>
<td>4</td>
</tr>
</tbody>
</table>

3.5.4 Predicting protein functions

This section reports the performance of the proposed TGCN in predicting “protein functions”.

Protein-to-protein interaction networks relate two proteins via multiple cell-dependent relations that can be modeled using multi-relational graphs; see Fig. 1.3. Protein classification seeks the unknown function of some proteins (nodes) based on the known functionality of a small subset of proteins and the protein-to-protein networks [148].

Given a target function $y_n$, that is known on a subset of proteins $n \in \mathcal{M}$, known functions on all proteins summarized in $X$, and the multi-relational protein networks $S$, the goal is to predict whether the proteins in $n \in \mathcal{V} - \mathcal{M}$ are associated with the target function or not. Hence, the number of target classes is $K = 2$. In this setting, $S_i$ represents the protein connectivity in the $i$th cell type which might be a cerebellum, midbrain, or frontal lobe cell. Table 3.3 summarizes the three datasets used in the following experiments.

We compare the TGCN with the GCN [80], which is the single-relational alternative, and Mune [141], which is a state-of-the-art diffusion-based approach for SSL over multi-relational graphs. Since GCN only accounts for a single graph, we select for the GCN the relation $i$ that achieves the best results in the validation set. Furthermore, Mune does not account for feature vectors in the nodes of the graph. Hence, to lay out a fair comparison, we employ the TGCN without using the feature vectors, i.e., $X = I_N$. Finally, since the classes are heavily unbalanced, we evaluate the performance of the various approaches using the macro F1 score for predicting the protein functions.

Fig. 3.8 report the macro F1 values for the aforementioned approaches for varying numbers of labeled samples $|\mathcal{M}|$. It is observed for all datasets that: i) the macro F1 score improves for increasing $|\mathcal{M}|$ across all algorithms ii) the TGCN that judiciously combines the multiple-relations outperforms the GCN by a large margin iii) For the case where nodal features are not

---

6Accurate classifiers achieve macro F1 values close to 1.
used (last two rows at each table), the TGCN outperforms the state-of-the-art Mune.

### 3.6 Conclusion

This chapter put forth a novel deep learning framework for SSL that utilized a tensor-graph architecture to sequentially process the input data. Experiments demonstrate the performance gains of the TGCN in the presence of noisy features, noisy edge weights, and random as well as adversarial edge perturbations.
Chapter 4

Efficient and Stable Graph Scattering Transforms via Pruning

The abundance of graph-abiding data calls for advanced learning techniques that complement nicely standard machine learning tools when the latter cannot be directly employed, e.g. due to irregular data inter-dependencies. Permeating the benefits of deep learning to graph data, graph convolutional networks (GCNs) offer a versatile and powerful framework to learn from complex graph data [17]. GCNs and variants thereof have been remarkably successful in social network analysis, 3D point cloud processing, recommender systems and action recognition. However, researchers have recently reported less consistent perspectives on the desirable GCN designs. For example, experiments in social network analysis have argued that deeper GCNs marginally increase the learning performance [132], whereas a method for 3D point cloud segmentation achieves state-of-the-art performance with a 56-layer GCN network [87]. These ‘controversial’ empirical findings motivate theoretical analysis to understand the fundamental performance-defining factors, and the resultant design choices for high-performance GCNs.

Aiming to bestow GCNs with theoretical guarantees, one promising path is to study graph scattering transforms (GSTs) – an analysis framework that has been advocated to assess stability and explain the success of deep neural networks (DNNs) [18, 93]. GSTs are non-trainable GCNs comprising a cascade of graph filter banks followed by nonlinear activation functions. The graph filter banks are designed analytically to scatter an input graph signal into multiple channels. GSTs extract stable features of graph data that can be utilized for downstream graph
learning tasks [43], with competitive performance especially when the number of training examples is small. Under certain conditions on the graph filter banks, GSTs are endowed with energy conservation properties [149], as well as stability that amounts to robustness to graph topology deformations [40]. Inherited from scattering transforms, GSTs however are known to incur exponential complexity in space and time that increases with the number of layers (GST depth) [18, 93]. Furthermore, stability should not come at odds with sensitivity. A filter’s output should be sensitive to and able to cope with perturbations of large magnitude. Current GST efforts have not addressed transform sensitivity to input noise. Lastly, graph data in different domains (e.g., social networks versus 3D point clouds) have distinct properties, which prompts domain-adaptive GST designs.

4.1 Related work

4.1.1 Graph convolutional networks

GCNs rely on a layered processing architecture comprising trainable graph convolutional operations to linearly combine features per graph neighborhood, followed by pointwise nonlinear functions applied to the linearly transformed features [17]. Complex GCNs and their variants have shown remarkable success in graph semi-supervised learning [80, 127] and graph classification tasks [143]. GCNs as simple as a single-layer linear module can offer high performance in certain social network learning applications [132]. On the other hand, a 56-layer GCN has
been employed to achieve state-of-the-art performance in 3D point cloud segmentation [87]. Whether simple or complex, designing GCNs guided by properties of the graph data at hand is of paramount importance.

### 4.1.2 Graph scattering transforms

To understand the success of GCNs analytically, recent works study the stability properties of GSTs with respect to metric deformations of the domain [40, 42, 149]. GSTs generalize scattering transforms [18, 93] to non-Euclidean domains. GSTs are a cascade of graph filter banks and nonlinear operations that is organized in a tree-structured architecture. The number of extracted GST features grows exponentially with the number of layers. Theoretical guarantees for GSTs are obtained after fixing the graph filter banks to implement a set of graph wavelets. The work in [149] establishes energy conservation properties for GSTs given that certain energy-preserving graph wavelets are employed, and also prove that GSTs are stable to graph structure perturbations; see also [42] that focuses on diffusion wavelets. On the other hand, [40] proves stability to relative metric deformations for a wide class of graph wavelet families. These contemporary works shed light into the stability and generalization capabilities of GCNs. However, stable transforms are not necessarily informative, and albeit highly desirable, a principled approach to selecting informative GST features remains still an uncharted venue.

### 4.1.3 Neural network based compression

This work advocates pruning of GSTs to avoid the exponential growth of features with the network depth, which is naturally related to deep neural network (DNN) based compression [51, 108]. State-of-the-art DNNs typically entail a large number of network layers and corresponding trainable parameters, which introduce excessive computational and memory costs [53, 85]. Confronting these challenges, neural network compression aims at reducing the network parameters and pruning the architecture to facilitate practical deployment of DNN-based solutions. Typical compression techniques prune redundant parameters during training, while minimizing the effect of the pruned parameters to the learned features [51, 108]. Recent approaches perform network pruning at initialization before training the DNN [86].

Although DNN-based compression is a fruitful direction, pruning GCNs has not received commensurate attention. Typical GCNs are applied to relatively small data sets and require only a
small number of layers and training parameters to attain state-of-the-art learning performance [80, 127, 132], which explains why GCN-based compression is yet to be explored. Deep GCNs are emerging however, to deal with web-scale graphs [87, 142]. Such overparameterized GCNs motivate the development of novel compression techniques. Our work can be seen as a stepping stone towards pruning GCNs; meanwhile, pruning GSTs versus GCNs/DNNs has differences in two aspects. First, GSTs are nontrainable feature extractors, whereas GCNs/DNNs introduce trainable parameters that should be considered by the compression algorithm. Second, GST pruning is inherently an one-shot process and pruning is performed in an online and even adaptive fashion, whereas pruning GCNs or DNNs should be performed offline.

4.1.4 Contributions

The present chapter develops a data-adaptive pruning approach to systematically retain informative GST features, which justifies the term pruned graph scattering transform (pGST). The pruning decisions are guided by a criterion promoting alignment (matching) of the input graph spectrum with that of the graph filters. The optimal pruning decisions are provided on-the-fly, and alleviate the exponential complexity of GSTs. We prove that the pGST is stable to perturbations of the input graph data and those of the network structure. Under certain conditions on the perturbation energy, the resultant pruning patterns before and after the network and input perturbations are identical, and the overall pGST is stable. Further, the sensitivity of pGST to random and localized noise is theoretically and experimentally investigated. It turns out that pGST is more sensitive to noise that is localized in the graph spectrum relative to noise that is uniformly spread over the spectrum. This is appealing because pGST can detect transient changes in the graph spectral domain, while ignoring small random perturbations. With extensive experiments we showcase that the proposed pGSTs perform similar to, and in certain cases better than, the baseline GSTs that use all scattering features, while achieving significant computational savings. Furthermore, the efficient and stable features extracted can be utilized towards graph classification and 3D point cloud recognition. Even without any training on the feature extraction step, the performance is comparable to state-of-the-art deep supervised learning approaches, particularly when training data are scarce. By analyzing the pruning patterns of the pGST, we deduce that graph data in different domains call for distinct network architectures; see Fig. 4.1. Finally, we establish that the number of pGST layers after pruning can be utilized to guide the design parameters of contemporary GCNs.
4.2 Background

Consider an undirected graph $G := \{V, E\}$ with node set $V := \{v_i\}_{i=1}^N$, and edge set $E := \{e_i\}_{i=1}^E$. Its connectivity is described by the graph shift matrix $S \in \mathbb{R}^{N \times N}$, whose $(n, n')$th entry $S_{nn'}$ is nonzero if $(n, n') \in E$ or if $n = n'$. A typical choice for $S$ is the symmetric adjacency or the Laplacian matrix. Further, each node can be also associated with a few attributes. Collect attributes across all nodes in the matrix $F := [f_1, \ldots, f_F] \in \mathbb{R}^{N \times F}$, where each column $f_f \in \mathbb{R}^N$ can be regarded as a graph signal.

4.2.1 Graph Fourier transform

A Fourier transform corresponds to the expansion of a signal over bases that are invariant to filtering; here, this graph frequency basis is the eigenbasis of the shift matrix $S$. Henceforth, $S$ is assumed normal with $S = V \Lambda V^\top$, where $V \in \mathbb{R}^{N \times N}$ forms the graph Fourier basis, and $\Lambda \in \mathbb{R}^{N \times N}$ is the diagonal matrix of corresponding eigenvalues $\lambda_0$, $\ldots$, $\lambda_{N-1}$, that can be thought of as graph frequencies. The graph Fourier transform (GFT) of $f \in \mathbb{R}^N$ is $\hat{f} = V^\top f \in \mathbb{R}^N$, while the inverse transform is $f = V \hat{f}$. The vector $\hat{f}$ represents the signal’s expansion in the eigenvector basis and describes the graph spectrum of $f$. The inverse GFT reconstructs the graph signal from its graph spectrum by combining graph frequency components weighted by the coefficients of the signal’s graph Fourier transform. GFT is a theoretical tool that has been popular for analyzing graph data in the graph spectral domain.

4.2.2 Graph convolutional networks

GCNs permeate the benefits of CNNs from processing Euclidean data to modeling graph structured data. GCNs model graph data through a succession of layers, each consisting of a graph convolutional operation (a.k.a. graph filter), a pointwise nonlinear function $\sigma(\cdot)$, and oftentimes also a pooling operation. Given $f \in \mathbb{R}^N$, the graph convolution operation diffuses each node’s information to its neighbors to obtain $Sf$ with $n$th entry $[Sf]_n = \sum_{n' \in N_n} S_{nn'} f_{n'}$ being a weighted average of the one-hop neighboring features. Successive application of $S$ will reach multi-hop neighbors, spreading the information across the network. Summing up, a
The $K$-th order graph convolutional operation (graph filtering) is

$$h(S)f := \sum_{k=0}^{K} w_k S^k f = V \hat{h}(\Lambda) \hat{f}$$

(4.1)

where the graph filter $h(\cdot)$ is parameterized by the learnable weights $\{w_k\}_{k=0}^{K}$, and in the graph spectral domain $\hat{h}(\Lambda) = \sum_{k=0}^{K} w_k \Lambda^k$. In the graph vertex domain, the learnable weights capture the influences from various orders of neighbors; and in the graph spectral domain, those weights adaptively adjust the focus and emphasize certain graph frequency bands. GCNs employ various graph filter banks per layer, and learn the parameters that minimize a predefined learning objective, such as classification, or regression.

### 4.2.3 Graph scattering transforms

GSTs are the training-free counterparts of GCNs, where the parameters of graph convolutions are fixed based on a design criterion. Per GST layer, the input is graph filtered using a filter bank $\{h_j(S)\}_{j=1}^{J}$, an elementwise nonlinear function $\sigma(\cdot)$, and a pooling operator $U$. At the first layer, the input $f \in \mathbb{R}^N$ constitutes the first scattering feature vector $z^{(0)} := f$. Next, $z^{(0)}$ is processed through $\{h_j(\cdot)\}_{j=1}^{J}$ and $\sigma(\cdot)$ to generate $\{z^{(j)}\}_{j=1}^{J}$ with $z^{(j)} := \sigma(h_j(S)z^{(0)})$. At the second layer, the same operation is repeated per $j$. This yields a tree structure with $J$ branches stemming out from non-leaf node; see also Fig. 4.2. The $\ell$-th layer of the tree includes $J^\ell$ nodes. Each node of layer $\ell$ is indexed by the path $p^{(\ell)}$ of the sequence of $\ell$ graph convolutions applied to the input $f$, i.e. $p^{(\ell)} := (j^{(1)}, j^{(2)}, \ldots, j^{(\ell)})$.

The scattering feature vector at the tree node indexed by $(p^{(\ell)}, j)$ at layer $\ell + 1$, is

$$z_{(p^{(\ell)}, j)} = \sigma(h_j(S)z_{(p^{(\ell)})})$$

(4.2)

where $p^{(\ell)}$ holds the list of indices of the parent nodes ordered by ancestry, and all path $p^{(\ell)}$ in the tree with length $\ell$ are included in the path set $\mathcal{P}^{(\ell)}$ with $|\mathcal{P}^{(\ell)}| = 2^\ell$. The memoryless nonlinearity $\sigma(\cdot)$ disperses the graph frequency representation through the spectrum, and endows the GST with increased discriminating power [40]. By exploiting graph sparsity, the computational complexity of (4.2) is $\mathcal{O}(KE)$, where $E = |E|$ is the number of edges in $G$. Each feature vector

---

1 A tree node is fully specified by its corresponding path.
2 Any analytical function $h(S)$ can be written as a polynomial of $S$ with maximum degree $N - 1$ [54].
Figure 4.2: Scattering pattern of a pGST with \( J = 3 \) and \( L = 3 \). Dashed lines represent pruned branches. An example \( f \) and GFTs of filter banks are depicted too. The third filter \( j = 3 \) at \( \ell = 1 \) is pruned because it generates no output (\( z(3) = 0 \)).

\( z_{(p(\ell))} \) is summarized by an aggregation operator \( U(\cdot) \) defining a scalar scattering coefficient \( \phi_{(p(\ell))} := U(z_{(p(\ell))}) \), where \( U(\cdot) \) is typically an averaging operator that reduces dimensionality of the extracted features. The scattering coefficient per tree node reflects the activation level at a certain graph frequency band.

These scattering coefficients are collected across all tree nodes to form a scattering feature map

\[
\Phi(f) := \left\{ \phi_{(p(\ell))} \right\}_{p(\ell) \in \mathcal{P}(\ell)}^{L}_{\ell=0}
\]

where \( |\Phi(f)| = \sum_{\ell=0}^{L} J^\ell \). The GST operation resembles a forward pass of a trained GCN. This is why several works study GST stability under perturbations of \( S \) in order to understand the working mechanism of GCNs [40, 42, 149].

4.3 Pruned Graph Scattering Transforms

While the representation power of GST increases with the number of layers, the computational and storage complexity of the transform also increase exponentially with the number of layers due to its scattering nature. Hence, even if informative features become available at deeper
layers, the associated exponential complexity of extracting such features is prohibitive. On the other hand, various input data (e.g., social networks and 3D point clouds) may have distinct properties, leading to different GST feature maps. In some cases, just a few nodes of deep layers are informative; and in other cases, nodes of shallow layers convey most of the information; see Fig. 4.1. This motivates the pursuit of a tuned GST to adaptively capture informative nodes.

Aspiring to improve GST models, we introduce a pruned (p)GST to systematically retain informative nodes without extra complexity. Our novel pGST reduces the exponential complexity and adapts GST to different graph data. Furthermore, pGST offers a practical mechanism to understand the architecture of GCNs. Based on the pruning patterns, the proposed pGST suggests when a deeper GCN is desirable, or, when a shallow one will suffice. Pruning the wavelet packets has been traditionally employed for compression in image processing applications [135], where the pruning is guided by a rate-distortion optimality criterion.

In this work, we consider a graph spectrum inspired criterion. Intuitively, each GST node will be associated with a unique subband of the graph spectrum. When the subband of a node does not have sufficient overlap with the graph signal spectrum, this node cannot capture the underlying graph signal, and should be pruned. Consider for specificity a smooth \( f \), where connected nodes have similar signal values. This implies sparse (low-rank) representation in the graph spectral domain; that is, \( \hat{f} := V^T f \in \mathbb{R}^N \) and \( [\hat{f}]_n = 0 \) for \( n \geq b \). The graph spectrum of the \( j \)th output is then

\[
V^T h_j(S)f = \text{diag}\left\{ (\hat{h}_j(\lambda)) \right\} \hat{f} = [\hat{h}_j(\lambda_1)\hat{f}_1, \hat{h}_j(\lambda)\hat{f}, \ldots, \hat{h}_j(\lambda_N)\hat{f}_N]^T
\]

(4.4)

where \( \lambda_n \) is the \( n \)th eigenvalue of \( S \) and each frequency \( \hat{f}_n \) is weighted by the corresponding transformed eigenvalue \( \hat{h}_j(\lambda_n) \). Hence, if the support of the spectrum \( \{\hat{h}_j(\lambda_n)\}_n \) is not included in the support of \( [\hat{f}]_n \), the \( j \)th graph filter output will not capture any information; that is, \( h_j(S)f = 0_N \); see Fig. 4.2. Thus, identifying such graph filters and pruning the corresponding tree nodes will result in a parsimonious and thus computationally efficient GST.

4.3.1 Pruning criterion

Motivated by this last observation, we introduce a pruning criterion to select the scattering branches per node by maximizing the alignment between the graph spectrum of the filters and
the scattering features. Per node \( p \), the optimization problem is

\[
\max_{\{f_{(p,j)}\}_{j=1}^{J}} \sum_{j=1}^{J} \left( \sum_{n=1}^{N} \left( \tilde{h}_j(\lambda_n)^2 - \tau \right) \left| \tilde{z}_{(p)} \right|^2 \right) f_{(p,j)} \quad (4.5)
\]

\[
\text{s. t.} \quad f_{(p,j)} \in \{0, 1\}, \quad j = 1, \ldots, J
\]

where \( \tilde{z}_{(p)} := Vz_{(p)} \) is the graph spectrum of the scattering feature vector \( z_{(p)} \); \( \tau \) is a user-specific threshold; and, \( f_{(p,j)} \) stands for the pruning assignment variable indicating whether node \( (p, j) \) is active \( (f_{(p,j)} = 1) \), or, it should be pruned \( (f_{(p,j)} = 0) \). The objective in \( 4.5 \) promotes retaining tree nodes that maximize the alignment of the graph spectrum of \( \tilde{z}_{(p)} \) with that of \( \tilde{h}_j(\lambda) \).

The threshold \( \tau \) introduces a minimum spectral value to locate those nodes whose corresponding graph spectral response is small, meaning \( \tilde{h}_j(\lambda_n)^2 \ll \tau \). Note that criterion \( 4.5 \) is evaluated per tree node \( p \), thus allowing for a flexible and scalable design.

The optimization problem in \( 4.5 \) is nonconvex since \( f_{(p,j)} \) is a discrete variable. Furthermore, recovering \( \tilde{z}_{(p)} \) requires an eigendecomposition of the Laplacian matrix that incurs complexity \( O(N^3) \). Nevertheless, by exploiting the structure in \( 4.5 \), we will develop an efficient pruning algorithm attaining the maximum of \( 4.5 \), as asserted by the following theorem.

**Theorem 4.1.** The optimal \( \{f^*_{(p,j)}\}_{j} \) are given by

\[
f^*_{(p,j)} = \begin{cases} 
1 & \text{if } \frac{\|z_{(p,j)}\|^2}{\|z_{(p)}\|^2} > \tau, \\
0 & \text{otherwise.}
\end{cases} \quad (4.6)
\]

The optimal variables \( f^*_{(p,j)} \) are given by comparing the energy of the input \( z_{(p)} \) to that of the output \( z_{(p,j)} \) per graph filter \( j \) that can be evaluated with linear complexity of order \( O(N) \). Our pruning criterion leads to a principled and scalable means of selecting the GST nodes to be pruned. The pruning objective is evaluated per node \( p \), and pruning decisions are made on-the-fly. Hence, when \( f^*_{(p)} = 1 \), node \( p \) is active and the graph filter bank will be applied to \( z_{(p)} \), expanding the tree to the next layer; otherwise, the GST will not be expanded further at node \( p \), which can effect exponential savings in computations. An example of such a pruned tree is depicted in Fig. 4.2. Evidently, the hyperparameter \( \tau \) controls the input-to-output energy ratio. A large \( \tau \) corresponds to an aggressively pruned scattering tree, while a small \( \tau \) amounts to a minimally pruned scattering tree. The stable and efficient representation extracted by the pGST...
is then defined as
\[
\Psi (f) := \{ \phi (p) \}_{p \in T}
\]
where \( T \) is the set of active tree nodes \( T := \{ p \in \mathcal{P} | f^*_p = 1 \} \).

Our pruning approach provides a concise version of GSTs and effects savings in computations as well as memory. Although the worst-case complexity of pGST is still exponential, a desirable complexity can be realized by properly selecting \( \tau \). As a byproduct, the scattering patterns of pGSTs reveal the appropriate depths and widths of the GSTs for different graph data; see also Fig. 4.1. The pruning approach so far is an unsupervised one, since the labels are not assumed available. Note that the GFT is used throughout for analytical purposes, and neither the pruning algorithm nor the pGST requires explicit calculation of the GFT.

### 4.3.2 Rate-distortion tradeoff

Our pruning objective is closely related to the optimal rate-distortion objective in signal processing \([107, 125]\). By judiciously selecting \( \tau \), the proposed pruning criterion in (4.6) finds also the optimal variables to the following problem

\[
\max_{\{ f(p,j) \}_{j=1}^J} \sum_{j=1}^J \frac{\| z(p,j) \|^2}{\| z(p) \|^2} f(p,j) \tag{4.7}
\]

s. t. \( f(p,j) \in \{ 0, 1 \}, \ j = 1, \ldots, J \)
\[
\sum_{j=1}^J f(p,j) \leq K
\]

where \( K \) denotes the maximum number of retained scattering features, that can be dictated by computational complexity constraints. The optimal solution to (4.7) is to set \( f(p,j) = 1 \) for the \( K \) scattering features with larger energy ratio \( \| z(p,j) \|^2 / \| z(p) \|^2 \) and the rest to zero \( f(p,j) = 0 \). Indeed, by properly choosing \( \tau \), the criterion in (4.6) can be employed to find the \( K \) channels with maximum energy, where \( \tau \) would be the \( K \)th larger energy ratio. Such energy preservation objective has been employed in signal processing in the context of PCA wavelets \([125]\). Indeed, the hyperparameter \( \tau \) reflects the sweet spot in the rate-distortion tradeoff \([107]\).
4.4 Stability and sensitivity of pGST

In this section, we study the stability and sensitivity of pGST when the input graph data and the network topology are perturbed. We will establish that pGST is stable in the presence of feature or network perturbations with bounded power. We will further analyze pGST sensitivity to input perturbations. To establish these results, we consider graph wavelets that form a frame with frame bounds $A$ and $B$, meaning for $f \in \mathbb{R}^N$, it holds that, $A^2 \|f\|^2 \leq \sum_{j=1}^{J} \|h_j(S)f\|^2 \leq B^2 \|f\|^2$. In the graph vertex domain, $A$ and $B$ characterize the numerical stability of recovering $f$ from $\{h_j(S)f\}_j$. In the graph spectral domain, they reflect the ability of a graph filter bank to amplify $f$ along each graph frequency. Tight frame bounds, satisfying $A^2 = B^2$, are of particular interest because such wavelets lead to enhanced numerical stability and faster computations [120].

The frame property of a graph wavelet plays an instrumental role in proving GST stability to perturbations of the underlying graph structure [40, 42, 149].

4.4.1 Stability to graph data perturbations

Perturbations present in $f$ may be attributed to modeling errors, or adversarial intervention aiming to poison learning. Consider the perturbed graph signal

$$\tilde{f} := f + \delta \in \mathbb{R}^N$$

(4.8)

where $\delta \in \mathbb{R}^N$ is the perturbation vector. We wish to study how and under what conditions our pGST is affected by such perturbations. A stable transformation should have a similar output under small input perturbations.

Before establishing that our pGST is stable, we first show that GST is stable to small perturbations in $f$. Prior art deals mainly with GST stability to structure perturbations [40, 42, 149].

**Lemma 4.1.** Consider the GST $\Phi(\cdot)$ with $L$ layers and $J$ graph filters; and suppose that the graph filter bank forms a frame with bound $B$, while $f$ and $\tilde{f}$ are related via (4.8). It then holds that

$$\frac{\|\Phi(f) - \Phi(\tilde{f})\|}{\sqrt{\|\Phi(f)\|}} \leq \sqrt{\sum_{\ell=0}^{L} (B^2 J)^\ell \sum_{\ell=0}^{L} J^\ell \|\delta\|}.$$  

(4.9)

The squared difference of the GSTs is normalized by the number of scattering features in
Φ(·), that is |Φ(f)| = \sum_{\ell=1}^{L} J^{\ell}. The bound in (4.9) relates to the frame bound of the wavelet filter bank. Clearly, for tight frames with B = 1, the normalized stability bound (4.9) is tight.

Let  \tilde{T}  be the structure of the pruned tree for  \Psi(\tilde{f}). The following lemma asserts that the pGST offers the same pruned tree for the original and the perturbed inputs.

Lemma 4.2. Let  \Psi(·)  denote the pGST with  L  layers and  J  graph filters;  \tilde{z}_{p}  the perturbed scattering feature at node  p; and,  \delta_{p} := z_{p} - \tilde{z}_{p}. If for all  p \in P  and  j = 1, \ldots, J, we have

\[ \|h_{j}(S)z_{p}\|^{2} - \tau \|z_{p}\|^{2} > \|h_{j}(S)\delta_{p}\|^{2} + \tau \|z_{p}\|^{2} - \|\tilde{z}_{p}\|^{2}. \]  

(4.10)

it then follows that

i) The pruned scattering transform will output the same tree for  \Psi(f)  and  \Psi(\tilde{f}); that is,  T = \tilde{T}; and,

ii) With  g(f) := \|h_{j}(S)f\|^{2} - \tau \|f\|^{2},  a necessary condition for (4.10) is

\[ |g(z_{p})| > g(\delta_{p}). \]  

(4.11)

According to (4.11), Lemma 4.2 can be interpreted as a signal-to-noise-ratio (SNR) condition because under  g(\delta_{p}) > 0,  it is possible to write (4.11) as  |g(z_{p})| / g(\delta_{p}) > 1. Lemma 4.2 provides a per-layer and branch condition for pGST to output the same pruned scattering tree for the original or the perturbed signal. This condition is also experimentally validated in Sec. 4.5 where the structure in  T  remains the same for Fig. 4.9b and Fig. 4.9a.

By combining Lemmas 4.1 and 4.2 we arrive at the following stability result for the pGST network.

Theorem 4.2. Consider the pGST transform  \Psi(·)  with  L  layers and  J  graph filters; and suppose that the graph filter bank forms a frame with bound  B, while  f  and  \tilde{f}  are related via (4.8). The pGST is stable to bounded perturbations  \delta,  in the sense that

\[ \frac{\|\Psi(f) - \Psi(\tilde{f})\|}{\sqrt{|\Psi(f)|}} \leq \sqrt{\sum_{\ell=0}^{L} F_{\ell} B^{2\ell}} / \sqrt{\sum_{\ell=0}^{L} F_{\ell}} \|\delta\| \]  

(4.12)

where  F_{\ell} := |P^{(\ell)} \cup T|  is the number of active scattering features at layer  \ell, and  |\Psi(f)| = \sum_{\ell=0}^{L} F_{\ell}  the number of retained scattering features.
The bound in (4.18) is linear in the perturbation power \( \| \delta \| \), and hence pGST is stable to perturbations in the input.

### 4.4.2 Stability to structural perturbations

We next study the effect of network perturbations to the proposed pGST. The goal here is to establish that pGST is also stable under structural perturbations that can be due to noise or adversarial attacks \([48, 150]\). Vanilla scattering transforms are invariant to translations and stable to perturbations that resemble translations \([18]\). Likewise, GSTs are invariant to permutations and stable to perturbations that are close to permutations \([40, 149]\). In the graph context, permutations are regarded as rearrangements of the vertex indices.

Consider a perturbed topology given by the \( N \times N \) perturbed (and permuted) shift matrix \( \tilde{S} \). The set of permutations that make \( \tilde{S} \) close to \( S \) are given by

\[
\mathcal{P}_0 := \arg \min_{P \in \mathcal{P}} \| P^\top \tilde{S} P - S \| \quad (4.13)
\]

where \( P \) is an \( N \times N \) permutation matrix having entries \( 0 - 1 \). Further, consider the following set of perturbation matrices \([40]\)

\[
\mathcal{D} := \left\{ \Delta : P^\top \tilde{S} P = S + \Delta^\top S + S \Delta, P \in \mathcal{P}_0 \right\} \quad (4.14)
\]

where \( \Delta \) denotes the \( N \times N \) perturbation matrix. The error is captured by the term \( \Delta^\top S + S \Delta \). Given the set of perturbation matrices in (4.14), we consider the distance between \( \tilde{S} \) and \( S \) as

\[
d(S, \tilde{S}) = \min_{\Delta \in \mathcal{D}} \| \Delta \|. \quad (4.15)
\]

Without loss of generality, let \( P = I \), otherwise fix a \( P_0 \in \mathcal{P} \) and define \( \tilde{S} \) to equal \( P_0^\top \tilde{S} P_0 \). Hence, the perturbed topology can be written as

\[
\tilde{S} = S + \Delta^\top S + S \Delta. \quad (4.16)
\]

Let \( \tilde{\Psi}(\cdot) \) denote the pruned scattering transform that is based on the perturbed topology \( \tilde{S} \), and \( \tilde{T} \) be the structure of the pruned tree for \( \tilde{\Psi}(f) \). The following lemma asserts that pGST outputs

---

3Unless it is stated otherwise the \( \ell_2 \) norms are employed.
the same pruned tree for both the original and the perturbed graph.

**Lemma 4.3.** Suppose $S$ and $\tilde{S}$ satisfy $d(S, \tilde{S}) \leq \varepsilon / 2$; and that for $\Delta \in \mathcal{D}$ with eigendecomposition $\Delta = U \text{diag} \{(d)\} U^\top$ it holds that $\|\Delta / d_{\text{max}} - I\| \leq \varepsilon$, where $d_{\text{max}}$ is the eigenvalue of $\Delta$ with maximum absolute value. Suppose the graph filter bank forms a frame with bound $B$, and $h(\lambda)$ satisfies the integral Lipschitz constraint $|\lambda h'(\lambda)| \leq C_0$. Let $\tilde{z}_p$ denote the perturbed scattering feature at the tree node $p$ and $\delta_p := z_p - \tilde{z}_p$. If for all nodes $p \in \mathcal{P}$ of layer $\ell$ and $j = 1, \ldots, J$, it holds that

$$\left|\left|h_j(S)z_p\right|^2 - \left\|z_p\right|^2\right| > (\ell \varepsilon C_0 B^{\ell-1} \|f\|)^2 + \tau \left\|\delta_p\right|^2 \quad (4.17)$$

the pruned scattering transform will then output the same tree for $\Psi(f)$ and $\tilde{\Psi}(f)$; that is, $\mathcal{T} = \tilde{\mathcal{T}}$.

Lemma 4.3 establishes conditions under which the pGST based on the original and perturbed graphs will output the same scattering tree. The assumption on the eigenvalues of the perturbation, that is $\|\Delta / d_{\text{max}} - I\| \leq \varepsilon$, limits the structural changes in the graph such as edge insertions or deletions; see also [40]. The integral Lipschitz constraint $|\lambda h'(\lambda)| \leq C_0$ is an additional stability requirement on the wavelet $h(\cdot)$. Note that these conditions are not required to establish stability to input noise in Theorem 4.2. By combining Lemma 4.3 with Proposition 3 in [40], we can show the stability of pGST to structural perturbations.

**Theorem 4.3.** Under the conditions of Lemma 4.3 it holds that

$$\frac{\|\Psi(f) - \tilde{\Psi}(f)\|}{\sqrt{\left|\Psi(f)\right|}} \leq \varepsilon C_0 \sqrt{\sum_{\ell=0}^{L} F_{\ell} \ell^2 B^{2\ell}} \|f\| \quad (4.18)$$

where $F_{\ell} := |\mathcal{P}(\ell) \cup \mathcal{T}|$ is the number of active scattering features at layer $\ell$, and $|\Psi(f)| = \sum_{\ell=0}^{L} F_{\ell}$ the number of retained scattering features.

The bound in (4.18) is linear in the perturbation $\varepsilon$, which establishes the stability of pGST. As the bound depends also linearly in the Lipschitz constant $C_0$ and the number of layers $L$, these terms also appear in the stability results for traditional GST [40]. For tight frames with $B = 1$, the bound in (4.18) can be further simplified. By contrasting the bounds in (4.12) and (4.18), we deduce that (4.12) is tighter than (4.18) by a factor $\ell$. This follows because the topology perturbation in (4.18) implies a perturbed wavelet that is repeatedly applied to the scattering
features. Hence, the error in (4.18) is accumulated layer by layer, giving rise to the $\ell$ factor in (4.18). On the other hand, for (4.12) the perturbation is introduced only at the first layer $\ell = 0$. Theorems 4.2 and 4.3 can be combined to guarantee stability under joint perturbations of both the input and the topology.

4.4.3 Sensitivity to graph data perturbations

Next, the sensitivity of pGST is analyzed for different types of noises in input signals. Although the stability of a transform is unquestionably important, pGST should be sensitive to signal perturbation that have some specific pattern. For example, signal perturbations may be distributed over all the nodes but are localized in the graph spectrum.

Specifically, two classes of input signal perturbations are considered, each having distinct graph spectral properties.

**Highly localized noise.** Here the energy of the perturbation signal is localized in the graph spectrum; that is,

$$E_L^{(\epsilon)} = \{ \delta_L \in \mathbb{R}^N | \tilde{\delta}_L = V \delta_L, \| \tilde{\delta}_L \| _0 \leq \epsilon, \| \tilde{\delta}_L \| _2 = 1 \}.$$  

(4.19)

**Random noise.** Here the energy of a perturbation signal is uniformly spread over the graph spectrum; that is,

$$E_R^{(\epsilon)} = \{ \delta_R \in \mathbb{R}^N | \tilde{\delta}_R = V \delta_R, [\tilde{\delta}_R]^2_\infty \leq \epsilon/N \}.$$  

(4.20)

The following corollary establishes the sensitivity bound for these classes of perturbation signals.

**Corollary 4.1.** Let $\Psi(\cdot)$ be a pGST with $L$ layers and $J = N$ filter. The $j$th filter has a transform function $h_j(S)$ with $\tilde{h}_j(\lambda_n) = 1$ for $n = j$, and $0$ for $n \neq j$. Consider two types of the perturbed inputs: $\tilde{f}_L := f + \delta_L$, where $\delta_L \in E_L$, and $\tilde{f}_R := f + \delta_R$, where $\delta_R \in E_R$. It then holds that

$$\| \Psi(f) - \Psi(\tilde{f}_R) \| \leq \sqrt{\left( \sum_{\ell=1}^{L} \frac{F_\ell B^{2\ell}}{N} + 1 \right) \epsilon}$$

---

The sensitivity analysis for structural perturbations does not provide interesting results; see the experimental validation in Section 4.5.3. Highly localized signal perturbation would affect only a few scattering filters; however, arbitrary structural perturbation would affect all scattering filters $\{h_j(S)\}_j$. Furthermore, different from data perturbations, the perturbed graph introduces errors at each GST layer. This makes the analysis non-trivial.
\[
\| \Psi(f) - \Psi(\tilde{f}_L) \| \leq \sqrt{\sum_{\ell=2}^{L} F^{\prime}_\ell B^{2\ell} + 2} \epsilon
\]

where \( F_\ell := |\mathcal{P}(\ell) \cup \mathcal{T}_R| \) and \( F^{\prime}_\ell := |\mathcal{P}(\ell) \cup \mathcal{T}_L| \) are the retained features in the cases of random and localized noise.

Corollary 1 suggests that pGST is more sensitive to \( \delta_L \) than to \( \delta_R \). This is appealing because pGST can detect transient changes in the graph spectral domain while ignoring small random perturbations. These results will be numerically corroborated in Sec. 4.5. Specifically, Fig. 4.9 shows that relative to the original \( \mathcal{T} \), the pruned tree changes considerably when the noise is localized in the input spectrum, while the tree remains the same for random noise.

### 4.5 Experiments

This section evaluates the performance of our pGST in various graph classification tasks. Graph classification amounts to predicting a label \( y_i \) given \( f_i \) and \( S_i \) for the \( i \)-th graph. Our pGST extracts the efficient and stable representation \( \Psi(f_i) \), which is utilized as a feature vector for predicting \( y_i \). During training, the structure of the pGST \( \mathcal{T} \) is determined, which is kept fixed during validation and testing. The parameter \( \tau \) is selected via cross-validation. Our goal is to provide tangible answers to the following research questions.

**RQ1** How does the proposed pGST compare to GST?

**RQ2** How does pGST compare to state-of-the-art GCNs in graph-based classification tasks?

**RQ3** Given graph data, what are pruned scattering patterns?

**RQ4** What is the impact of signal perturbations in the scattering patterns?

**RQ5** Can the effective GCN depth linked with the pGST depth?

#### 4.5.1 Pruned GST compared to traditional GST

To address RQ1, we reproduce the experiments of two tasks in [40]: authorship attribution and source localization. For the scattering transforms, we consider three implementations of graph
Figure 4.3: Classification accuracy against number of samples in the authorship attribution (a) and SNR in dB for source localization (b). The percentage of features after pruning retained in the top-2\(|\mathcal{T}|\) most important GST features given by the SVM classifier (c).

Figure 4.4: Runtime comparison of the scattering transforms for Fig. 4.3.
filter banks: the diffusion wavelets (DS) in [42], the monic cubic wavelets (MCS) in [50] and the tight Hann wavelets (THS) in [120]. The scattering transforms use $J = 5$ filters, $L = 5$ layers, and $\tau = 0.01$. The extracted features from GSTs are subsequently utilized by a linear support vector machine (SVM) classifier.

Authorship attribution amounts to determining if a certain text was written by a specific author. Each text is represented by a graph with $N = 244$, where words (nodes) are connected based on their relative positions in the text, and $f$ is a bag-of-words representation of the text; see also [42]. Fig. 4.3 (a) reports the classification accuracy as the number of training samples (texts) increases. GSTs utilize $\sum_{\ell=1}^{5} 5^\ell = 781$ scattering coefficients, while pGSTs rely only on $|T| = 61$ for $P_{DS}$, $|T| = 30$ for $P_{MCS}$, and $|T| = 80$ for $P_{THS}$. Evidently, pGST achieves comparable performance as the baseline GST, whereas pGST uses only a subset of features (12.8%, 3.8% and 10.2%, respectively). The SVM classifier provides a coefficient that weighs each scattering scalar. The magnitude of each coefficient shows the importance of the corresponding scattering feature in the classification. Fig. 4.3 (c) depicts the percentage of features retained after pruning in the top-2$|T|$ most important GST features given by the SVM classifier. It is observed, that although pGST does not take into account the labels, the retained features are indeed informative for classification.

Source localization amounts to recovering the source of a rumor given a diffused signal over a Facebook subnetwork with $N = 234$; see the detailed settings in [42]. Fig. 4.3 (b) shows the classification accuracy of the scattering transforms as the SNR (in dB) increases. In accordance to Lemma 1 and Theorem 2, both pGST and GST are stable over a wide range of SNR. Furthermore, the performance of pGST matches that of GST, while the pGST uses only a subset of features. Finally, Fig. 4.4 depicts the runtime of the different scattering approaches, where the computational advantage of the pruned methods is evident.

Ablation study

Fig. 4.5 reports how the pGST is affected by varying the threshold $\tau$ in the task of source localization, with $J = 6$ and $L = 5$. Fig. 4.5 (a) shows the classification accuracy that generally decreases as $\tau$ increases since the number of active features $|T|$ decreases; cf. Fig. 4.5 (b). Fig. 4.5 (c) reports the runtime in seconds. Fig. 4.6, 4.7 showcase the classification performance of pGST with $\tau = 0.01$ for varying $L$ with $J = 3$, and for varying $J$ with $L = 3$, respectively. It

---

$p_{DS}$, $p_{MCS}$, $p_{THS}$ denote the pruned versions of these transforms.
Figure 4.5: Performance of pGSTs for varying $\tau$.

Figure 4.6: Classification accuracy over $L$

Figure 4.7: Classification accuracy over $J$
is observed, that the classification performance generally improves with \( L \) and \( J \).

### 4.5.2 Pruned GST for graph-based classification tasks

In response to RQ2, we consider three graph-based classification tasks: graph classification, graph-based 3D point cloud classification, and semi-supervised node classification.

#### Graph classification

We compare the proposed pGST with the following state-of-the-art methods: The kernel methods shortest-path \([16]\), and Weisfeiler-Lehman optimal assignment (WL-OA) \([84]\); the deep learning approaches PatchySan \([99]\), GraphSage \([49]\), edge-conditioned filters in CCNs (ECC) \([122]\), Set2Set \([128]\), SortPool \([145]\), and DiffPool \([143]\); and the geometric scattering classifier (GSC) \([43]\). Results are presented on the protein data sets D&D, Enzymes and Proteins, and the scientific collaboration data set Collab. The parameters of these datasets are listed in Table 4.1. Since the Collab dataset did not have nodal features, \( f \) was selected as the vector that holds the node degrees. We performed 10-fold cross validation and report the classification accuracy averaged over the 10 folds. The gradient boosting classifier is employed for pGST and GST with parameters chosen based on the performance on the validation set. The graph scattering transforms use the MC wavelet with \( L = 5 \), \( J = 5 \), and \( \tau = 0.01 \). Table 4.2 lists the classification accuracy of the proposed and competing approaches. Even without any training on the feature extraction step, the performance of pGST is comparable to the state-of-the-art deep supervised learning approaches across all datasets. GST and pGST outperform also GSC, since the latter uses a linear SVM to classify the scattering features.

\[\text{Table 4.1: Dataset characteristics}\]

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Graphs</th>
<th>Features ( F )</th>
<th>Max ( N ) per graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collab</td>
<td>5000</td>
<td>1</td>
<td>492</td>
</tr>
<tr>
<td>D&amp;D</td>
<td>1178</td>
<td>89</td>
<td>5748</td>
</tr>
<tr>
<td>Enzymes</td>
<td>600</td>
<td>3</td>
<td>126</td>
</tr>
<tr>
<td>Proteins</td>
<td>1113</td>
<td>3</td>
<td>620</td>
</tr>
</tbody>
</table>

For the competing approaches we report the 10-fold cross-validation numbers reported by the original authors; see also \([143]\).
Table 4.2: Graph classification accuracy.

<table>
<thead>
<tr>
<th>Method</th>
<th>Data Set</th>
<th>ENZYMES</th>
<th>D&amp;D</th>
<th>COLLAB</th>
<th>PROTEINS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Kernel</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHORTEST-PATH</td>
<td></td>
<td>42.32</td>
<td>78.86</td>
<td>59.10</td>
<td>76.43</td>
</tr>
<tr>
<td>WL-OA</td>
<td></td>
<td>60.13</td>
<td>79.04</td>
<td>80.74</td>
<td>75.26</td>
</tr>
<tr>
<td>PATCHYSAN</td>
<td></td>
<td>–</td>
<td>76.27</td>
<td>72.60</td>
<td>75.00</td>
</tr>
<tr>
<td>GRAPHSAGE</td>
<td></td>
<td>54.25</td>
<td>75.42</td>
<td>68.25</td>
<td>70.48</td>
</tr>
<tr>
<td>ECC</td>
<td></td>
<td>53.50</td>
<td>74.10</td>
<td>67.79</td>
<td>72.65</td>
</tr>
<tr>
<td><strong>GNNs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set2Set</td>
<td></td>
<td>60.15</td>
<td>78.12</td>
<td>71.75</td>
<td>74.29</td>
</tr>
<tr>
<td>SortPool</td>
<td></td>
<td>57.12</td>
<td>79.37</td>
<td>73.76</td>
<td>75.54</td>
</tr>
<tr>
<td>DiffPool-Det</td>
<td></td>
<td>58.33</td>
<td>75.47</td>
<td>82.13</td>
<td>75.62</td>
</tr>
<tr>
<td>DiffPool-NoLP</td>
<td></td>
<td>62.67</td>
<td>79.98</td>
<td>75.63</td>
<td>77.42</td>
</tr>
<tr>
<td>DiffPool</td>
<td></td>
<td><strong>64.23</strong></td>
<td>81.15</td>
<td>75.50</td>
<td>78.10</td>
</tr>
<tr>
<td><strong>Scattering</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GSC</td>
<td></td>
<td>53.88</td>
<td>76.57</td>
<td>76.88</td>
<td>74.03</td>
</tr>
<tr>
<td>GST</td>
<td></td>
<td>59.84</td>
<td>79.28</td>
<td>77.32</td>
<td>76.23</td>
</tr>
<tr>
<td>PGST (Ours)</td>
<td></td>
<td>60.25</td>
<td><strong>81.27</strong></td>
<td>78.40</td>
<td><strong>78.57</strong></td>
</tr>
</tbody>
</table>

Figure 4.8: 3D point cloud classification. (a) 9843 clouds for training and 2468 for testing (b) 615 clouds for training and 11703 for testing
**Point cloud classification**

We further test pGST in classifying 3D point clouds. Given a point cloud, a graph can be created by connecting points (nodes) to their nearest neighbors based on their Euclidian distance. Each node is also associated with 6 scalars denoting its x-y-z coordinates and RGB colors. For this experiment, GSTs are compared against PointNet++ [104, 105], 3dShapeNets [134] and VoxNet [96], that are state-of-the-art deep learning approaches. Fig. 4.8 reports the classification accuracy for the ModelNet40 dataset [134] for increasing $L$. In Fig. 4.8 (a) 9,843 clouds are used for training and 2,468 for testing using the gradient boosting classifier; whereas, in Fig. 4.8 (b) only 615 clouds are used for training and the rest for testing using a fully connected neural network classifier with 3 layers. The scattering transforms use an MC wavelet with $J = 5$ for Fig. 4.8 (a), and $J = 9$ for Fig. 4.8 (b). Fig. 4.8 showcases that scattering transforms are competitive to state-of-the-art approaches, while pGST outperforms GST. This may be attributed to overfitting effects, since a large number of GST features is not informative. Furthermore, the exponential complexity of GSTs prevents their application with $L = 6$. Fig. 4.8 (b) shows that when the training data are scarce, GST and pGST outperform the PointNet++, which requires a large number of training data to optimize over the network parameters.

**Semi-supervised node classification**

A task of major importance at the intersection of machine learning and network science is semi-supervised learning (SSL) over graphs. Given the topology $S$, the features in the $N \times F$ matrix $F$, and labels only at a subset $L$ of nodes $\{y_n\}_{n \in L}$ with $L \subset V$, the goal of is to predict the labels $\{y_n\}_{n \in U}$ of the unlabeled set of nodes. For this task, we utilize the pGST algorithm to extract the feature vectors $\{z(p)\}_{p \in T}$ from $F$, which are then utilized by a fully connected neural network with 3 layers to predict the missing labels. To facilitate comparison, we reproduce the experimental setup of [80], namely the same split of the data for training, validation, and testing sets, and compare the various methods in the Cora dataset with $N = 2,708$ nodes, $C = 7$ classes and $F = 1,433$ features. The pGST algorithm employs the TH wavelet with $L = 5$, $J = 3$, and $\tau = 0.01$. We also compare to [149], where the GST is employed, the full scattering coefficients are extracted and then dimensionality reduction is effected using PCA to handle the large number of features. The so obtained features are processed by a fully connected network. Table 4.3 reports the classification accuracy of several state-of-the-art methods. Our method
Table 4.3: SSL classification accuracy.

<table>
<thead>
<tr>
<th>Method</th>
<th>Cora</th>
</tr>
</thead>
<tbody>
<tr>
<td>ManiReg [12]</td>
<td>59.5</td>
</tr>
<tr>
<td>SemiEmb [131]</td>
<td>59.0</td>
</tr>
<tr>
<td>LP [146]</td>
<td>68.0</td>
</tr>
<tr>
<td>Planetoid [140]</td>
<td>75.7</td>
</tr>
<tr>
<td>GCN [80]</td>
<td>81.5</td>
</tr>
<tr>
<td>GAT [127]</td>
<td>83.0</td>
</tr>
<tr>
<td>GST [149]</td>
<td>81.9</td>
</tr>
<tr>
<td>pGST (ours)</td>
<td>81.9</td>
</tr>
</tbody>
</table>

that performs unsupervised feature extraction is highly competitive and outperforms all except the GAT approach. It is worth noting that pGST performs comparably to the GST approach in [149], which suggests that our spectrum-inspired criterion has a ‘PCA-like’ effect. However, PCA needs all the scattering features to process them offline, while pGST prunes the scattering features on-the-fly, and allows for increased scalability.

4.5.3 Scattering patterns analysis

To address RQ3, we depict the scattering structures of pGSTs, with an MC wavelet, \( J = 3 \), and \( L = 5 \), for the Collab, Proteins, and ModelNet40 datasets in Fig. 4.1. Evidently, graph data from various domains require an adaptive scattering architecture. Specifically, most tree nodes for the academic collaboration dataset are pruned, and hence most informative features reside at the shallow layers. This is consistent with the study in [132], which experimentally shows that deeper GCNs do not contribute as much for social network data. These findings are further supported by the small-world phenomenon in social networks, which suggests that the diameter of social networks is small [130]. On the other hand, the tree nodes for a 3D point cloud are minimally pruned, which is in line with the work in [87] that showcases the advantage of deep GCNs in 3D point clouds classification.

Sensitivity of the pGST tree to signal perturbations

Next, we evaluate noise effects in the pruning algorithm to provide an answer to RQ4. Corollary 1 suggests that localized noise will have a more prominent effect on the pGST than random noise. This is appealing because pGST can detect transient changes in the graph spectral domain, while...
Figure 4.9: The pGST applied to the Cora dataset with and without signal perturbation, corroborating the result of Corollary 1.

Sensitivity of the pGST tree to structural perturbations

For the same setting as Fig. 4.9, the effect of the structural perturbations in the structure of pGST is also tested. The adjacency matrix was perturbed by adding noise in the eigenvalues such that the SNR is -20dB. Random noise in this scenario means that the noise is distributed over the eigenvalues, whereas localized noise affects only a subset of the eigenvalues significantly. Fig. 4.10 shows that the localized and random noise do not alter the scattering patterns significantly. This is in accordance with the stability results in Theorem 3.
Finally, we empirically explore the connections between PGST and GCN design, in response to RQ5. We validate the scattering patterns in Fig. 1 for the Protein datasets. The validation employs DiffPool that is a state-of-the-art GCN; see [143].

Fig. 4.11 depicts the pruned scattering patterns for the three protein datasets (Enzyme, Protein and DD). Most connections after $l = 5$ are pruned. This suggests that for protein data $L = 5$ graph convolution layers capture most information.

Fig. 4.12 shows the performance of DiffPool for these datasets as the number of GCN layers increases. The performance of DiffPool does not improve significantly for more than 5 GCN layers, which corroborates the insights obtained from the pGST patterns.
4.6 Conclusion

This chapter developed a novel approach to pruning the graph scattering transform. Experiments demonstrated i) the performance gains of pGSTs relative to GSTs; ii) that pGST is competitive in a variety of graph classification tasks; and (iii) graph data from different domains exhibit unique pruned scattering patterns, which calls for adaptive network architectures.
Chapter 5

Summary and Future Directions

Backed by rigorous theoretical and extensive experimental results, the present thesis has introduced novel algorithms that help realize the goal of robust deep learning for graph data. The following subsections provide a summary of the work presented in this thesis, as well as possible future research directions.

5.1 Thesis Summary

In order to provide a framework to effectively detect anomalous nodes in graphs, Chapter 2 introduced a graph-based random sampling and consensus approach. Rigorous analysis provides performance guarantees for the proposed algorithm, by bounding the number of random draws involved. GraphSAC outperforms competing algorithms in detecting random walk-based anomalies, clustered anomalies, as well as contemporary adversarial attacks for graph data. GraphSAC has also been adapted to robustify the performance of state-of-the-art SSL methods. Experiments demonstrate that GraphSAC significantly enhances the performance of contemporary deep learning methods.

Chapter 3 put forth a TGCN architecture for robust deep learning over multir-relational networks. The approach is computationally affordable, since the number of operations scales linearly with respect to the number of graph edges. Instead of committing a fortiori to a specific type of diffusion, the TGCN learns the diffusion pattern that best fits the data. TGCN can be adapted to robustify SSL over a single graph with model-based, adversarial or random edge perturbations. To account for adversarial perturbations, an ED module was developed that first
performed random dithering to the (nominal) graph edges and then used the dithered graphs as input to the TGCN.

Finally, Chapter 4 introduced a pruning framework for graph scattering transforms. The proposed pGST relies on a graph-spectrum-based data-adaptive criterion to prune less informative features on-the-fly, and effectively reduce the computational complexity of GSTs. Stability of pGST is established in the presence of perturbations in the input or the network structure. Sensitivity analysis of pGST reveals that the transform is more sensitive to noise that is localized in the graph spectrum relative to noise that is uniformly spread over the spectrum, which is appealing since pGST can detect transient changes in the graph spectral domain. The novel pGST extracts efficient and stable features of graph data as demonstrated in extensive experiments.

5.2 Future Research

The promising results in this thesis open up interesting directions for a number of future research topics. The following subsections discuss a few of these directions.

5.2.1 GraphSAC for regression

In Chapter 3 GraphSAC was designed to cope with detection of perturbed categorical nodal values. Adapting GraphSAC to accommodate real-valued perturbed $y_n$ attributes is well motivated for learning in new environments. Our initial approach here will be to replace the SSL model with a kernel-based inference algorithm; see e.g. [59, 66, 67, 110, 111, 123].

Per realization $r$, under S1 we will collect the real-valued samples in an $L \times 1$ vector $y^{(r)}$ that will be used to solve the optimization problem (cf. S2)

$$\hat{y}^{(r)} = \arg\min_y \|y^{(r)} - S^{(r)} y\|^2_2 + y^T K^{-1} y$$

where the $L \times N$ matrix $S^{(r)}$ selects the entries of $y$ corresponding to sampled nodes, and $K$ denotes the $N \times N$ Laplacian kernel matrix. Depending on desirable graph properties such as diffusion rate or smoothness of $y$, Laplacian kernels are different functions of $A$ [123].

The reconstructed $\hat{y}^{(r)}$ in (5.1) is available in closed form. The updated consensus set is $\mathcal{C}^{(r)} := \{ n \in \mathcal{U}^{(r)} : (y_n - \hat{y}_n^{(r)})^2 < \epsilon \}$, where $\epsilon$ is an appropriate threshold. Averaging across realizations, $\hat{y}$ will be found as $\hat{y} = \sum_{r=1}^R \hat{y}^{(r)} b^{(r)} / \sum_{r=1}^R b^{(r)}$, with $b^{(r)}$ as in S3. The
anomalous nodes will then be revealed as those with the largest deviation \((y_n - \hat{y}_n)^2\). Besides the preliminary plan outlined here, we can experiment with different distance functions such as the \(\ell_1\) cost that is robust to outliers, and several regularizers effecting sparsity, low-rank, or total variation constraints.

### 5.2.2 Dynamic GraphSAC

Unveiling anomalies over time and among nodes will bring benefit in various contemporary applications. Building on our recent works dealing with inference over dynamic graphs \cite{45,64,66,109}, we will broaden the proposed GraphSAC framework of Chapter 2 to account for dynamic graphs and attributes. Dynamic GraphSAC will be developed to accommodate temporal as well as graph anomalies. The modular design of the proposed GraphSAC allows in principle to replace the SSL algorithm in S2 with an algorithm tailored for inference of spatio-temporal processes evolving over networks. Consider the time-evolving vector \(y_t\) that may represent the fuel or ammo of a warfighter at time \(t\) and the \(L \times 1\) vector \(\bar{y}^{(r)}_t\) holding the values of sampled nodes at draw \(r\). To adapt GraphSAC for detecting spatio-temporal anomalies, we will consider the following formulation

\[
\min_{\{y^r\}_{r=1}^t} \sum_{\tau=1}^t \|\bar{y}^{(r)}_\tau - S^{(r)}y_\tau\|_2^2 + \mu_1 \sum_{\tau=1}^t \|y_\tau - A^{(\tau,\tau-1)}y_{\tau-1}\|_2^2 K^{(\chi)}_{\tau} + \mu_2 \sum_{\tau=1}^t \|y_\tau\|_2^2 K^{(\nu)}_{\tau} \tag{5.2}
\]

where the \(L \times N\) matrix \(S^{(r)}\) selects the entries of \(y_t\) corresponding to sampled nodes, \(K^{(\chi)}_{\tau}\) and \(K^{(\nu)}_{\tau}\) are \(N \times N\) Laplacian kernel matrices, and \(A^{(\tau,\tau-1)}\) can be viewed as a graph transition matrix. The second sum in (5.2) effects smoothness over time and space by choosing an appropriate spatio-temporal kernel \(K^{(\chi)}_{\tau}\); while the third sum promotes smoothness of \(y_t\) over the connected nodes of the graph via \(K^{(\nu)}_{\tau}\). Although (5.2) is a batch objective, building on our approach in \cite{63,64}, we will pursue an online solver such as that offered by kernel kriged Kalman filtering that in addition to Markovian dynamics, it comes with affordable complexity. Given the per-draw estimates \(\{\hat{y}^{(r)}_\tau\}_{\tau}\), the remaining steps will follow the lines of GraphSAC-based regression. Thorough tests and rigorous analysis will aim at assessing the performance of our proposed approach to discover and track anomalous nodes across space and time.
5.2.3 Robust dynamic GCNs

Deep learning over dynamically evolving networks will bring benefit to multiple applications. This prompts us to explore integrating recurrent neural networks (RNNs) into the proposed robust TGCN of Chapter 3. The envisioned adaptive graph recurrent neural network (AGRN) will receive time-evolving inputs $X_t$ and $G_t$, and output labels $Y_t$. The ED module for dynamic graphs may also utilize a slow-varying constraint [6]. We will start with $f(\tilde{Z}_t^{(l)}, \theta_t^{(l)})$, where $\theta_t^{(l)}$ are trainable parameters. The proposed AGRN will employ the following linear operation at time $t$ and layer $l$

$$Z_t^{(l)} := f(\tilde{Z}_t^{(l-1)}, \theta_t^{(l)}) + f(\tilde{Z}_{t-1}^{(l-1)}, \theta_{t-1}^{(l-1)})$$

(5.3)

where $\tilde{Z}_{t-1}^{(l-1)}$ is the hidden representation from the previous time instant and $\theta_{t-1}^{(l)}$ are the corresponding trainable parameters. The optimization objective will be augmented with regularizers promoting temporal smoothness. The weights of the novel AGRN will be learned using backpropagation through time.

5.2.4 Low-rank and sparse identification of anomalies

In Chapter 3, we account for perturbed edges via the randomized edge dithering scheme. However, we could attempt to prepossess the perturbed adjacency matrix and identify the anomalous edges. Typically adversaries alter only a small subset of edges. This suggests a prepossessing step to learn a “base” adjacency matrix $B$ from the perturbed $\tilde{A}$. Matrix $B$ can be estimated as

$$\min_{S, B > 0} \|\tilde{A} - S - B\|_F^2 + \lambda_\nu\|B\|_\nu^2 + \lambda_1\|S\|_1 + \text{tr}(X^T BX),$$

(5.4)

where $\lambda_\nu, \lambda_1 > 0$ are coefficients obtained via cross validation. The nuclear norm in (5.4) promotes low rank in $B$; the entry-wise $\ell_1$ norm encourages the solution for $S$ to be sparse; and the Laplacian regularization promotes feature smoothness with respect to the estimated adjacency. Intuitively, the low-rank base matrix $B$ captures common patterns, that is graph communities, while $S$ is a first step to capture spurious connections in $\tilde{A}$. Our AGCN will then use $B$ as a template for generating the $I$ ED auxiliary graphs $\{G_i\}_{i=1}^I$. This preprocessing step will reduce the search space of our AGCN, thus boosting computational efficiency. A preliminary version of this work can be found in [103].
To further robustify our preprocessing step, we will leverage robust community detection \cite{103,117,118} to account for the egonets of the perturbed graph $\tilde{A}$ in (5.4). Stacking the adjacency matrices of the egonets as slabs, we will form the tensor perturbed adjacency $\tilde{A}$, and postulate its decomposition into a low-rank tensor $B$ plus a sparse tensor $S$; see also \cite{11,94,121}. Our idea then is to identify the wanted tensor summands by solving (cf. (5.4))

$$\min_{S,B,U,V,W \geq 0} \|\tilde{A} - S - B\|_F^2 + \lambda_s \|B - [[U, V, W]]\|_F^2 + \lambda_1 \|S\|_1 + \sum_{n=1}^{N} \operatorname{tr}(X^T B_n X)$$

(5.5)

where $[[U, V, W]] := \sum_{k=1}^{K} u_k \circ v_k \circ w_k$, denotes a canonical polyadic decomposition with factor matrices $U := [u_1, \ldots, u_K], V := [v_1, \ldots, v_K], W := [w_1, \ldots, w_K]$ as in e.g., \cite{121}, and $B_n$ is the $n$th frontal slab of $B$. The effectiveness of the proposed preprocessing step will be quantified both analytically and experimentally, and additional tensor regularizers will be investigated.
References


[38] G. W. Flake, S. Lawrence, C. L. Giles et al., “Efficient identification of web communities,” in Proc. Intl. Conf. on Knowledge Disc. and Data Mining (KDD), Boston, MA, USA, Aug. 2000, pp. 150–160.


Appendix A

Proofs for Chapter 2

A.1 Proof of Theorem 1

First, a link between the GraphSAC sampling scheme (2.5) and the GraphSAC filter step in S3 is established. Using the law of total probability (l.t.p), the total probability of sampling any contaminated subset is

$$\sum_{\mathcal{L} \in \tilde{\mathcal{L}}_L^c} p_G(\mathcal{L}) = |\tilde{\mathcal{L}}_L^c| f.$$  \hfill (A.1)

with $\tilde{\mathcal{L}}_L := \{\mathcal{L} \subseteq \mathcal{V}, |\mathcal{L}| = L, \mathcal{L} \cap A = \emptyset\}$ the set containing subsets without anomalous nodes, while the complementary set $\tilde{\mathcal{L}}_L^c$ contains all the remaining size-$L$ subsets $\tilde{\mathcal{L}}_L^c := \mathcal{L}_L \setminus \tilde{\mathcal{L}}_L$. This probability coincides with the probability of declaring all contaminated samples as clean. Using the l.t.p. and the uniform sampling strategy it holds that

$$\sum_{\mathcal{L} \in \tilde{\mathcal{L}}_L} \Pr(b = 1|\mathcal{L} \in \tilde{\mathcal{L}}_L^c) \Pr(\mathcal{L} \in \tilde{\mathcal{L}}_L^c)$$

$$= |\tilde{\mathcal{L}}_L^c| \Pr(b = 1|\mathcal{L} \in \tilde{\mathcal{L}}_L^c) \Pr(\mathcal{L} \in \tilde{\mathcal{L}}_L^c)$$  \hfill (A.2)

where $\Pr(\mathcal{L} \in \tilde{\mathcal{L}}_L^c) = |\tilde{\mathcal{L}}_L^c|/|\mathcal{L}_L|$ due to the uniform sampling. By introducing the probability of false alarm as $P_{fa} := \Pr(b = 1|\mathcal{L} \in \tilde{\mathcal{L}}_L^c)$ and equating the right side in (A.1) and (A.2) it holds
that

\[ P_{fa} := \frac{|L_L|}{|L_L'|} f \]  \hfill (A.3)

Next, using the l.t.p. it holds for \( p_G(L) \) \hfill (2.10) that

\[ 1 = |\tilde{L}_L|d + |\tilde{L}_c^L|f \]  \hfill (A.4)

\[ d = \frac{1 - |\tilde{L}_c^L|f}{|L_L|} \]  \hfill (A.5)

Furthermore by the definition of \( P \), it follows that

\[ P = \mathbb{E}_{L^{(r)} \sim p_G}[\hat{P}^{(r)}] \]  \hfill (A.6)

\[ = \sum_{L^{(r)} \in L} p_G(L^{(r)}) \hat{P}^{(r)} \]  \hfill (A.7)

\[ = \sum_{L^{(r)} \in \tilde{L}_L} \hat{P}^{(r)} d + \sum_{L^{(r)} \in \tilde{L}_c^L} \hat{P}^{(r)} f \]  \hfill (A.8)

\[ = d|\tilde{L}_L|P_{nom} + f|\tilde{L}_c^L|P_{anom} \]  \hfill (A.9)

where (A.8) holds since \( \tilde{L}_L \) and \( \tilde{L}_c^L \) are disjoint subsets, (A.9) follows by definition of \( P_{nom} = 1/|\tilde{L}_L| \sum_{L^{(r)} \in \tilde{L}_L} \hat{P}^{(r)} \) and \( P_{anom} = 1/|\tilde{L}_c^L| \sum_{L^{(r)} \in \tilde{L}_c^L} \hat{P}^{(r)} \). Using (A.4) and (A.3) it holds that

\[ P = (1 - \frac{|\tilde{L}_c^L|^2}{|L_L'|} P_{fa}) P_{nom} + \frac{|\tilde{L}_c^L|^2}{|L_L'|} P_{fa} P_{anom}. \]  \hfill (A.10)

Hence, \( P \) is a convex combination between the nominal label distribution \( P_{nom} \) and the one affected by the anomalies \( P_{anom} \). By using (A.10) the result of Theorem 1 follows

\[ \|P - P_{nom}\|_1 = \frac{|\tilde{L}_c^L|^2}{|L_L'|} P_{fa}\|P_{anom} - P_{nom}\|_1. \]  \hfill (A.11)

For the cardinality of the involved sets of subsets counting theory will be used. Specifically, it follows

\[ |\tilde{L}_L| = \binom{N - K}{L} \]  \hfill (A.12)
since this is the number of ways to choose $L$ nodes out of $N - K$. Similarly, by counting the subsets with size $L$ constructed out of $N$ nodes it holds that

$$|\mathcal{L}_L| = \binom{N}{L}.$$  \hfill (A.13)

Finally, since the subsets $\bar{\mathcal{L}}_L^c$ and $\bar{\mathcal{L}}_L$ are not overlapping it holds that $|\bar{\mathcal{L}}_L^c| = |\mathcal{L}_L| - |\bar{\mathcal{L}}_L|$ and

$$|\bar{\mathcal{L}}_L^c| = \binom{N}{L} - \binom{N - K}{L}. \hfill (A.14)$$

### A.2 Proof of Theorem 2

First, by the l.t.p., $\mathbb{E}_{\mathcal{L}(r)}[\hat{\mathbf{P}}^{(r)} b^{(r)}]$ is written as

$$\mathbb{E}_{\mathcal{L}(r)}[\hat{\mathbf{P}}^{(r)} b^{(r)}] = \mathbb{E}_{\mathcal{L}(r)}[\hat{\mathbf{P}}^{(r)} b^{(r)} | \mathcal{L}(r) \in \bar{\mathcal{L}}_S] p_{\bar{\mathcal{L}}_S}$$

$$+ \mathbb{E}_{\mathcal{L}(r)}[\hat{\mathbf{P}}^{(r)} b^{(r)} | \mathcal{L}(r) \notin \bar{\mathcal{L}}_S] (1 - p_{\bar{\mathcal{L}}_S}) \hfill (A.15)$$

where $p_{\bar{\mathcal{L}}_S} := \Pr(\mathcal{L} \in \bar{\mathcal{L}}_S)$. Next, by applying the assumption $\Pr(b^{(r)} = 1 | \mathcal{L}(r) \cap A = \emptyset) = 1$ it holds that

$$\mathbb{E}_{\mathcal{L}(r)}[\hat{\mathbf{P}}^{(r)} b^{(r)} | \mathcal{L}(r) \in \bar{\mathcal{L}}_S] = \mathbb{E}_{\mathcal{L}(r), \mathcal{L}(r) \cap A = \emptyset}[\hat{\mathbf{P}}^{(r)}] = \mathbf{P}_{\text{nom}}. \hfill (A.17)$$

Furthermore, by using the assumption $\Pr(b^{(r)} = 0 | \mathcal{L}(r) \cap A > K_m) = 1$ it follows that

$$\mathbb{E}_{\mathcal{L}(r)}[\hat{\mathbf{P}}^{(r)} b^{(r)} | \mathcal{L} \notin \bar{\mathcal{L}}_S] = \hfill (A.18)$$

$$\mathbb{E}_{\mathcal{L}(r)}[\hat{\mathbf{P}}^{(r)} b^{(r)} | \mathcal{L} \in \mathcal{L}_S^{K_m}] \Pr(\mathcal{L} \in \mathcal{L}_S^{K_m} | \mathcal{L} \notin \bar{\mathcal{L}}_S) \hfill (A.19)$$

The equality in (A.18) implies that GraphSAC always discards samples corrupted with at least $K_m + 1$ adversaries. Hence, by using (A.17) and (A.18), it follows from (A.15) that

$$\mathbb{E}_{\mathcal{L}(r)}[\hat{\mathbf{P}}^{(r)} b^{(r)}] = \mathbf{P}_{\text{nom}} p_{\bar{\mathcal{L}}_S} + \mathbb{E}_{\mathcal{L}(r)}[\hat{\mathbf{P}}^{(r)} b^{(r)} | \mathcal{L} \in \mathcal{L}_S^{K_m}] p_{\mathcal{L}_S^{K_m}} \hfill (A.20)$$
where \( P^{K_m}_{L_S} = \Pr(\mathcal{L} \in \mathcal{L}_{S}^{K_m} | \mathcal{L} \notin \tilde{\mathcal{L}}_S)(1 - P_{L_S}) = \Pr(\mathcal{L} \in \mathcal{L}_{S}^{K_m}) \). Furthermore, since \( P_G = \mathbb{E}_{L^{(r)}}[\hat{P}^{(r)}b^{(r)}/\mathbb{E}_{L}[b^{(r)}]] \) with \( P_{\delta} := \mathbb{E}_{L}[b^{(r)}] \) then \( ||P_G - P_{\text{nom}}||_1 \) can be written as

\[
\| \frac{1}{P_{\delta}} (P_{\text{nom}}P_{L_S} + \mathbb{E}_{L^{(r)}}[\hat{P}^{(r)}b^{(r)}|\mathcal{L}^{(r)} \in \mathcal{L}_{S}^{K_m}]P_{L_S}^{K_m}) - P_{\text{nom}} \|_1 \tag{A.21}
\]

Using the l.t.p. \( \mathbb{E}_{L^{(r)}}[\hat{P}^{(r)}b^{(r)}|\mathcal{L}^{(r)} \in \mathcal{L}_{S}^{K_m}] \) is written as

\[
\mathbb{E}_{L^{(r)}}[\hat{P}^{(r)}b^{(r)}|\mathcal{L}^{(r)} = 1, \mathcal{L} \in \mathcal{L}_{S}^{K_m}] \Pr(b^{(r)} = 1|\mathcal{L}^{(r)} \in \mathcal{L}_{S}^{K_m}) = P_{\text{fa}} \tag{A.23}
\]

where \( P_{fa} := \Pr(b^{(r)} = 1|\mathcal{L}^{(r)} \notin \tilde{\mathcal{L}}_S) = \Pr(b^{(r)} = 1|\mathcal{L}^{(r)} \in \mathcal{L}_{S}^{K_m}) \) by assumption. Following a similar argument it holds for \( P_{\delta} \) that

\[
P_{\delta} = \Pr(b^{(r)} = 1|\mathcal{L}^{(r)} \in \tilde{\mathcal{L}}_S)p_{L_S}
+ \Pr(b^{(r)} = 1|\mathcal{L}^{(r)} \notin \tilde{\mathcal{L}}_S)(1 - p_{L_S}) \tag{A.25}
\]

\[
p_{L_S} + \Pr(b^{(r)} = 1|\mathcal{L}^{(r)} \in \mathcal{L}_{S}^{K_m})(1 - p_{L_S}) \tag{A.26}
\]

\[
p_{L_S} + P_{fa}(1 - p_{L_S}) \tag{A.27}
\]

where (A.25) follows from the l.t.p. and (A.26) is a result of applying \( \Pr(b^{(r)} = 1|\mathcal{L}^{(r)} \in \tilde{\mathcal{L}}_S) = 1 \) and \( \Pr(b^{(r)} = 0|\mathcal{L}^{(r)} \cap \mathcal{A} > K_m) = 1 \). Hence, the result in Theorem 2 is follows.

### A.3 Proof of Theorem 3

In this section \( ||P||_1 \) represents the norm 1 of matrix \( P \), and not the sum of absolute values of the elements of \( P \). Notice that since each draw of nodes \( \mathcal{L}^{(r)} \) is independent, \( \hat{P}^{(r)} \) is also independent across \( r \). For each \( \hat{P}^{(r)} \) it further holds

\[
||\hat{P}^{(r)}||_1 \leq \sqrt{||\hat{P}^{(r)}||_1||\hat{P}^{(r)}||_\infty} \leq \sqrt{N} \tag{A.28}
\]

where the last inequality follows since \( [\hat{P}^{(r)}]_{n,c} \in [0, 1], \forall n, c \) and \( \sum_{c=1}^{C}[\hat{P}^{(r)}]_{n,c} = 1, \forall n \) that implies \( ||\hat{P}^{(r)}||_1 \leq 1 \) and \( ||\hat{P}^{(r)}||_\infty \leq N \). Next, the second moment of the matrix \( \hat{P}^{(r)} \) is also
bounded

\[ m_2(\hat{P}^{(r)}) = \max \left[ \| \mathbb{E} \{ \hat{P}^{(r)} \hat{P}^{(r)^\top} \} \|, \| \mathbb{E} \{ \hat{P}^{(r)} \hat{P}^{(r)^\top} \} \| \right]. \] (A.29)

Let us define \( M = \mathbb{E} \{ \hat{P}^{(r)^\top} \hat{P}^{(r)} \} \). For the spectral norm it holds that

\[ \| M \| \leq \sqrt{\| M \|_{\infty} \| M \|_1} \leq N \] (A.30)

where the last inequality holds since \( M_{n,n'} = \mathbb{E} \{ \sum_{c=1}^C [\hat{P}^{(r)}]_{n,c} [\hat{P}^{(r)}]_{n',c} \} \leq 1 \) since \( [\hat{P}^{(r)}]_{n,c} \in [0,1], \forall n,c \) and \( \sum_{c=1}^C [\hat{P}^{(r)}]_{n,c} = 1, \forall n \), and thus \( \| M \|_{\infty} \leq N \) and \( \| M \|_1 \leq N \). Similarly, it can be shown that \( \| \mathbb{E} \{ \hat{P}^{\top} \hat{P} \} \| \leq N \). Furthermore, the matrix variance is bounded as

\[ v(\hat{P}) \leq N \] (A.31)

Hence, by appealing to the matrix Bernstein inequality for the sum of uncentered random matrices [124] (2.14) and (2.15) in Theorem 2 follow.

**A.4 Proof of Corollary 1**

For the following consider that \( Y_L \) is an \( N \times C \) matrix whose \( n \)-th row is \( y_n^\top \) if \( n \in \mathcal{L} \) and is \( 0_L^\top \) otherwise. First, for diffusion-based SSL models it holds that

\[ \| P_{\text{nom}} - P_{\text{anom}} \|_1 = \| h(A)(\frac{1}{|L_L|} \sum_{L \in \mathcal{L}_L} Y_L - \frac{1}{|L'_L|} \sum_{L' \in \mathcal{L}'_L} Y_{L'}) \|_1 \] (A.32)

If \( n \in \mathcal{N} \) is nominal node, where \( \mathcal{N} = \mathcal{V} - \mathcal{A} \), then \( n \) is contained in \( {N-K-L-1 \choose L-1} \) subsets of \( \mathcal{L}_L \) since \( n \) may participate in a subset of size \( L - 1 \) with any of the \( N - K - 1 \) remaining nominal nodes. Hence, each row \( n \) of \( Y \) in the first sum of (A.32) is added \( {N-K-L-1 \choose L-1} \) times if \( n \in \mathcal{N} \) and 0 if \( n \in \mathcal{A} \). Hence, it follows

\[ \frac{1}{|L_L|} \sum_{L \in \mathcal{L}_L} Y_L = \frac{{N-K-L-1 \choose L-1}}{{N-K \choose L}} Y_{\mathcal{N}} \] (A.33)

\[ = \frac{L}{N-K} Y_{\mathcal{N}} \] (A.34)
For an anomalous node, \( n \in \mathcal{A} \) is contained in \( f_A := \binom{N-1}{L-1} \) subsets of \( \tilde{\mathcal{L}}_L^c \) since \( n \) may participate in a subset of size \( L - 1 \) with any of the \( N - 1 \) remaining nodes. On the contrary, for a nominal node \( n \in \mathcal{N} \), the number of times \( f_N \), \( n \) appears in subsets of \( \tilde{\mathcal{L}}_L^c \) can not be computed with straightforward counting theory, since any subset containing \( n \) must contain at least one anomalous node. However, the total number of nodes in all subsets of \( \tilde{\mathcal{L}}_L^c \) is expressed as

\[
S|\tilde{\mathcal{L}}_L^c| = Kf_A + (N - K)f_N \tag{A.35}
\]

where the left side of (A.35) follows since there are \(|\tilde{\mathcal{L}}_L^c|\) subsets each containing \( L \) nodes and the right side of (A.35) holds since there are \( K(N - K) \) anomalous (nominal) nodes each included in \( f_A(f_N) \) subsets of \( \tilde{\mathcal{L}}_L^c \). Hence, it holds that

\[
f_N = \frac{L}{N - K}|\tilde{\mathcal{L}}_L^c| - \frac{K}{N - K}f_A. \tag{A.36}
\]

Hence, the \( n \)-th row of \( \mathbf{Y} \) in the second sum of (A.32) is added \( f_A \) times if \( n \in \mathcal{A} \) and \( f_N \) times if \( n \in \mathcal{N} \). Thus, it follows

\[
\frac{1}{|\mathcal{L}_L^c|} \sum_{\mathcal{L}' \in \mathcal{L}_L^c} \mathbf{Y}_{\mathcal{L}'} = \frac{f_N}{|\mathcal{L}_L^c|} \mathbf{Y}_N + \frac{f_A}{|\mathcal{L}_L^c|} \mathbf{Y}_A \tag{A.37}
\]

By utilizing (A.34) and (A.37) it holds that

\[
\frac{1}{|\mathcal{L}_L^c|} \sum_{\mathcal{L} \in \mathcal{L}_L^c} \mathbf{Y}_{\mathcal{L}} - \frac{1}{|\mathcal{L}_L^c|} \sum_{\mathcal{L}' \in \mathcal{L}_L^c} \mathbf{Y}_{\mathcal{L}'}
\]

\[
= \frac{L}{N - K} \mathbf{Y}_N - \frac{f_N}{|\mathcal{L}_L^c|} \mathbf{Y}_N - \frac{f_A}{|\mathcal{L}_L^c|} \mathbf{Y}_A \tag{A.38}
\]

\[
= \frac{Kf_A}{|\mathcal{L}_L^c|((N - K))} \mathbf{Y}_N - \frac{f_A}{|\mathcal{L}_L^c|} \mathbf{Y}_A \tag{A.39}
\]

\[
= \frac{f_A}{|\mathcal{L}_L^c|} \left( \frac{K}{N - K} \mathbf{Y}_N - \mathbf{Y}_A \right) \tag{A.40}
\]

Hence, for diffusion-based SSL models it holds that

\[
\| \mathbf{P}_{\text{nom}} - \mathbf{P}_{\text{anom}} \|_1 = \| \frac{f_A}{|\mathcal{L}_L^c|} h(A) \left( \frac{K}{N - K} \mathbf{Y}_N - \mathbf{Y}_A \right) \|_1 \tag{A.41}
\]
Furthermore, using canonical vectors $Y_N$ is written as

$$Y_N = \sum_{n \in N} e_n y_n^\top$$  \hspace{1cm} (A.42)

where $e_n$ is a canonical vector with 1 at the $n$-th position and otherwise 0. Hence, if $h_n$ is the $n$-th column of $h(A)$ it follows from (A.41) that

$$\|P_{\text{nom}} - P_{\text{anom}}\|_1 = f_A \frac{K}{N - K} \sum_{n \in N} h_n y_n^\top - \sum_{n \in A} h_n y_n^\top \|_1.$$  \hspace{1cm} (A.43)
Appendix B

Proofs for Chapter 4

B.1 Proof of Theorem 1

By its definition, the objective in (4.5) can be rewritten as

\[
J \sum_{j=1}^{J} \left( \sum_{n=1}^{N} \left( \hat{h}_j(\lambda_n)^2 - \tau \right) [\hat{z}(p)]_n^2 \right) f_j
\]

\[
= \sum_{j=1}^{J} \hat{z}_j^T(p) \left( \text{diag} \left\{ \left( \hat{h}_j(\lambda) \right) \right\}^2 - \tau I \right) \hat{z}_j(p) f_j .
\]

By introducing the scalars \( \alpha_j := \hat{z}_j^T(p) \left( \text{diag} \left\{ \left( \hat{h}_j(\lambda) \right) \right\}^2 - \tau I \right) \hat{z}_j(p) \) for \( j = 1, \ldots, J \), (4.5) can be rewritten as

\[
\max_{f_j} \sum_{j=1}^{J} \alpha_j f_j \tag{B.1}
\]

s. t. \( f_j \in \{0, 1\}, \quad j = 1, \ldots, J \).

The optimization problem in (B.1) is nonconvex since \( f_j \) is a discrete variable. However, maximizing the sum in (B.1) amounts to setting \( f_j = 1 \) for the positive \( \alpha_j \) over \( j \). Such an
approach leads to the optimal pruning assignment variables
\[ f^*_j = \begin{cases} 
1 & \text{if } \alpha_j > 0, \\
0 & \text{if } \alpha_j < 0.
\end{cases} \quad (B.2) \]

The rest of the proof focuses on rewriting \( \alpha_j \) as
\[ \alpha_j = \hat{z}^T(p) \text{diag} \{ \hat{h}_j(\lambda) \}^2 - \tau I \hat{z}(p) \quad (B.3) \]
\[ = \| \text{diag} \{ \hat{h}_j(\lambda) \} \hat{z}(p) \|^2 - \tau \| \hat{z}(p) \|^2 \quad (B.4) \]

Furthermore, since matrix \( V \) is orthogonal, it holds that \( \| \hat{z}(p) \|^2 = \| V^T z(p) \|^2 = \| z(p) \|^2 \) from which it follows that
\[ \| \text{diag} \{ \hat{h}_j(\lambda) \} \hat{z}(p) \|^2 = \| h_j(S) z(p) \|^2 \quad (B.5) \]
\[ = \| \sigma(h_j(S) z(p)) \|^2 \]
\[ = \| h_j(S) z(p) \|^2 \quad (B.6) \]

where the second line follows because \( \sigma(\cdot) \) is applied elementwise, and does not change the norm.

### B.2 Proof of Lemma 1

By the definition in (4.3), it holds that
\[ \| \Phi(x) - \Phi(\tilde{x}) \|^2 = \sum_{\ell=0}^{L} \sum_{\mu(\ell) \in \mathcal{P}(\ell)} \| \phi(\mu(\ell)) - \tilde{\phi}(\mu(\ell)) \|^2 \quad (B.7) \]

which suggests bounding each summand in (B.7) as
\[ | \phi(\mu(\ell)) - \tilde{\phi}(\mu(\ell)) | = | U(z(\mu(\ell)) - U(\tilde{z}(\mu(\ell))) | \]
\[ \leq \| U \| \| z(\mu(\ell)) - \tilde{z}(\mu(\ell)) \| \quad (B.8) \]
\[ \leq \| U \| \| z(\mu(\ell)) - \tilde{z}(\mu(\ell)) \| \quad (B.9) \]
where (B.9) follows since the norm is a sub-multiplicative operator. Next, we will show the recursive bound

\[
\|z_{(p(\ell))} - \tilde{z}_{(p(\ell))}\| = \|\sigma(h_{j(\ell)}(S)z_{(p(\ell-1))}) - \sigma(h_{j(\ell)}(S)\tilde{z}_{(p(\ell-1))})\| \leq \|\sigma()\| \|h_{j(\ell)}(S)z_{(p(\ell-1))} - h_{j(\ell)}(S)\tilde{z}_{(p(\ell-1))}\| \leq \|h_{j(\ell)}(S)\| \|z_{(p(\ell-1))} - \tilde{z}_{(p(\ell-1))}\| (B.10)
\]

where (B.11), (B.13) hold because the norm is a sub-multiplicative operator, and (B.12) follows since the nonlinearity is nonexpansive, i.e. \(\|\sigma()\| < 1\). Hence, by applying (B.13) \(\ell - 1\) times, the following condition holds

\[
\|z_{(p(\ell))} - \tilde{z}_{(p(\ell))}\| \leq \|h_{j(\ell)}(S)\| \|h_{j(\ell-1)}(S)\| \cdots \|h_{j(1)}(S)\| \|x - \bar{x}\| (B.14)
\]

and by further applying the frame bound and (4.8), we deduce that

\[
\|z_{(p(\ell))} - \tilde{z}_{(p(\ell))}\| \leq B^\ell \|\delta\| (B.15)
\]

Combining (B.9), (B.15) and the average operator property \(\|U\| = 1\) it holds that

\[
|\phi_{(p(\ell))} - \tilde{\phi}_{(p(\ell))}| \leq B^\ell \|\delta\| (B.16)
\]

By applying the bound (B.16) for all entries in the right hand side of (B.7) it follows that

\[
\|\Phi(x) - \Phi(\bar{x})\|^2 \leq \sum_{\ell=0}^L \sum_{p(\ell) \in P(\ell)} B^{2\ell} \|\delta\|^2 (B.17)
\]

By factoring out \(\|\delta\|\) and observing that the sum in the right side of (B.17) does not depend on the path index \(p\) it follows that

\[
\|\Phi(x) - \Phi(\bar{x})\|^2 \leq \left(\sum_{\ell=0}^L |P(\ell)| B^{2\ell}\right) \|\delta\|^2 (B.18)
\]
Finally, since the cardinality of the paths at $\ell$ is $|\mathcal{P}(\ell)| = J^\ell$ and $\sum_{\ell=0}^L (B^2 J)^\ell = ((B^2 J)^L) / (B^2 J - 1)$ it holds

$$\|\Phi(x) - \Phi(\tilde{x})\| \leq \sqrt{\frac{(B^2 J)^L}{B^2 J - 1}} \|\delta\| \quad \text{(B.19)}$$

### B.3 Proof of Lemma 2

We will prove the case for $\ell = 0$, where $z_{j(0)} = x$, since the same proof holds for any $\ell$. First, we adapt (4.10) to the following

$$\|h_j(S)x\|^2 - \tau \|x\|^2 > \|h_j(S)\delta\|^2 + \tau \|x\|^2 - \|\tilde{x}\|^2. \quad \text{(B.20)}$$

The proof will examine two cases and will follow by contradiction. For the first case, consider that branch $j$ is pruned in $\Psi(x)$ and not pruned in $\Psi(\tilde{x})$, i.e. $(j) \notin T$ and $(j) \in \tilde{T}$. By applying (4.6) for $z_{(j)} = \sigma(h_j(S)x)$ there exists $C \geq 0$ such that

$$\frac{\|h_j(S)x\|^2}{\|x\|^2} \leq \tau - C \quad \text{(B.21)}$$

$$\|h_j(S)x\|^2 \leq \tau \|x\|^2 - C \|x\|^2 \quad \text{(B.22)}$$

Furthermore, from (4.6) it holds for $\tilde{z}_{(j)} = \sigma(h_j(S)\tilde{x})$ that

$$\frac{\|h_j(S)\tilde{x}\|^2}{\|\tilde{x}\|^2} > \tau \quad \text{(B.23)}$$

By applying (4.8) to (B.23), and using the triangular inequality it follows that

$$\|h_j(S)x\|^2 + \|h_j(S)\delta\|^2 \geq \tau \|\tilde{x}\|^2 \quad \text{(B.24)}$$

Next, by applying (B.22) it holds that

$$\tau \|x\|^2 - C \|x\|^2 + \|h_j(S)\delta\|^2 \geq \tau \|\tilde{x}\|^2 \quad \text{(B.25)}$$

$$\tau(\|x\|^2 - \|\tilde{x}\|^2) + \|h_j(S)\delta\|^2 \geq C \|x\|^2. \quad \text{(B.26)}$$
Next, by utilizing (B.20) and the absolute value property \(|a| \geq a\) to upper-bound the left side of (B.26) it follows that

\[
\|h_j(S)x\|^2 - \tau\|x\|^2 > C\|x\|^2. \tag{B.27}
\]

Finally, by applying (B.22) the following is obtained

\[
0 > 2C\|x\|^2 \tag{B.28}
\]

which implies that \(C < 0\). However, this contradicts (B.21) since \(C \geq 0\). Following a similar argument we can complete the proof for the other case.

### B.4 Proof of Lemma 3

The proof of Lemma 3 requires the following result.

**Lemma B.1.** Consider the shift matrix \(S\) and the perturbed matrix \(\tilde{S}\), such that \(d(S, \tilde{S}) \leq \mathcal{E}/2\). Further, for \(\Delta \in \mathcal{D}\) with eigendecomposition \(\Delta = U \text{diag} \{(d)\} U^\top\) it holds that \(\|\Delta/d_{\max} - I\| \leq \mathcal{E}\), where \(d_{\max}\) is the eigenvalue of \(\Delta\) with maximum absolute value. At layer \(\ell\) consider the scattering feature indexed by \(p^{(\ell)}\) of the original transform \(\Psi(x)\) as \(z_{(p^{(\ell)})}\) and of the perturbed transform \(\tilde{\Psi}(x)\) as \(\tilde{z}_{(p^{(\ell)})}\). Suppose also that graph filter bank forms a frame with bound \(B\), and \(h(\lambda)\) satisfies the integral Lipschitz constraint \(|\lambda h'(\lambda)| \leq C_0\). It then holds that

\[
\|\tilde{z}_{(p^{(\ell)})} - z_{(p^{(\ell)})}\| \leq \mathcal{E}C_0B^{\ell-1}\|x\|. \tag{B.29}
\]

**Proof.** First, we add and subtract on (B.29) the term \(\sigma(h_{j^{(\ell)}}(S))\sigma(h_{j^{(\ell-1)}}(\tilde{S}) \ldots \sigma(h_{j^{(1)}}(\tilde{S})x) \ldots)\)

\[
\|\tilde{z}_{(p^{(\ell)})} - z_{(p^{(\ell)})}\| \tag{B.30}
\]

\[
\leq \|\sigma(h_{j^{(\ell)}}(S))\sigma(h_{j^{(\ell-1)}}(\tilde{S}) \ldots \sigma(h_{j^{(1)}}(\tilde{S})x) \ldots) - \sigma(h_{j^{(\ell)}}(S))\sigma(h_{j^{(\ell-1)}}(S) \ldots \sigma(h_{j^{(1)}}(S)x) \ldots)\|
\]

\[
- \sigma(h_{j^{(\ell)}}(S))\sigma(h_{j^{(\ell-1)}}(S) \ldots \sigma(h_{j^{(1)}}(S)x) \ldots)
\]

\[
+ \sigma(h_{j^{(\ell)}}(\tilde{S}))\sigma(h_{j^{(\ell-1)}}(\tilde{S}) \ldots \sigma(h_{j^{(1)}}(\tilde{S})x) \ldots)\|
\]

\[
\leq \|\sigma(h_{j^{(\ell)}}(S))\sigma(h_{j^{(\ell-1)}}(\tilde{S}) \ldots \sigma(h_{j^{(1)}}(\tilde{S})x) \ldots)\|. \tag{B.31}
\]
\[
-s(h_{j(\ell)}(S)s(h_{j(\ell-1)}(S)\cdots s(h_{j(1)}(S)x)\cdots)) \\
+ \|s(h_{j(\ell)}(S)s(h_{j(\ell-1)}(\tilde{S})\cdots s(h_{j(1)}(\tilde{S})x)\cdots)) \\
-s(h_{j(\ell)}(\tilde{S})s(h_{j(\ell-1)}(\tilde{S})\cdots s(h_{j(1)}(\tilde{S})x)\cdots))\|\quad (B.32)
\]

By a similar argument as in (B.14), it holds for the first summand in (B.32) that
\[
\|s(h_{j(\ell)}(S)s(h_{j(\ell-1)}(\tilde{S})\cdots s(h_{j(1)}(\tilde{S})x)\cdots)) \\
-s(h_{j(\ell)}(\tilde{S})s(h_{j(\ell-1)}(\tilde{S})\cdots s(h_{j(1)}(\tilde{S})x)\cdots))\| \\
\leq B\|\tilde{z}_{(p(\ell-1))} - z_{(p(\ell-1))}\|\quad (B.33)
\]

For the second summand in (B.32), we consider the following
\[
\|z_{(p(\ell))}\| \leq \|h_{j(\ell)}(S)\|\|h_{j(\ell-1)}(S)\|\cdots\|h_{j(1)}(S)\|\|x\| \\
\leq B^\ell\|x\| \quad (B.34)
\]

which can be derived following (B.14). Also by utilizing the result of Proposition 2 in [40] it can be shown that
\[
\|h_{j}(S) - h_{j}(\tilde{S})\| \leq \epsilon C_0. \quad (B.35)
\]

where \( C_0 \) is the integral Lipschitz constant \( |\lambda h'(\lambda)| \leq C_0 \). By combining (B.33)-(B.35) we arrive to the following recursive condition
\[
\|\tilde{z}_{(p(\ell))} - z_{(p(\ell))}\| \leq \|\tilde{z}_{(p(\ell-1))} - z_{(p(\ell-1))}\| + (\ell - 1)\epsilon C_0 B^{\ell-1}\|x\|
\]
which can be solved to arrive at (B.29). \( \square \)

Now that Lemma 4 is established we proceed with the proof for Lemma 3 follows. The proof will examine two cases and will follow by contradiction. For the first case, consider that branch \( j \) is pruned in \( \Psi(x) \) and not pruned in \( \tilde{\Psi}(x) \), i.e. \( (j) \not\in T \) and \( (j) \in \tilde{T} \). By applying (4.6) for \( z_{(p,j)} = \sigma(h_{j}(S)z_{(p)}) \) there exists \( C \geq 0 \) such that
\[
\frac{\|h_{j}(S)z_{(p)}\|^2}{\|z_{(p)}\|^2} \leq \tau - C \quad (B.36)
\]
\[ \| h_j(S)z_p \|^2 \leq \tau \| z_p \|^2 - C \| z_p \|^2 \] (B.37)

Furthermore, from (4.6) it holds for \( \tilde{z}_{(p,j)} = \sigma(h_j(S)\tilde{z}_{(p)}) \) that

\[ \frac{\| h_j(\tilde{S})\tilde{z}_{(p)} \|^2}{\| \tilde{z}_{(p)} \|^2} > \tau \] (B.38)

By using the triangle inequality to (B.38), we obtain

\[ \| h_j(S)z_p \|^2 + \| h_j(\tilde{S})\tilde{z}_{(p)} - h_j(S)z_p \|^2 \geq \tau \| \tilde{z}_{(p)} \|^2 \] (B.39)

and upon applying (B.37), we arrive at

\[ \tau (\| z_p \|^2 - \| \tilde{z}_{(p)} \|^2) + \| h_j(\tilde{S})\tilde{z}_{(p)} - h_j(S)z_p \|^2 \geq C \| z_p \|^2. \] (B.40)

Next, we leverage Lemma 4 and the absolute value property \( |a| \geq a \) to upper-bound the left side of (B.40) as

\[ \tau (\| z_p \|^2 - \| \tilde{z}_{(p)} \|^2) + \left( \ell \| x \| \right)^2 > C \| z_p \|^2. \] (B.41)

By applying (4.17) in Lemma 3, we deduce that

\[ \| h_j(S)z_p \|^2 - \| z_p \|^2 > C \| z_p \|^2. \] (B.42)

Finally, by applying (B.37), we find

\[ 0 > 2C \| z_p \|^2 \] (B.43)

which implies that \( C < 0 \). However, this contradicts (B.36) since \( C \geq 0 \). Following a symmetric argument, we can complete the proof for the other case too.