Towards Understanding and Avoiding Limitations of Convolutions on Graphs

Dissertation

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Andreas Roth

Dortmund

2025

Dekan:	Prof. Dr. Jens Teubner
Gutachter*innen:	Jun.-Prof. Dr. Thomas Liebig
	Assoz. Prof. Dr. Nils Morten Kriege

Abstract

Many challenging tasks involve data with an inherent graph structure. Examples of such graph-structured data include molecules, road networks, and transaction records. As such data becomes increasingly available, the need for methods that can effectively leverage graph-structured data grows rapidly. While message-passing neural networks (MPNNs) have shown promising results, their impact on real-world applications remains limited. Although various phenomena have been identified as possible limitations to the performance of MPNNs, their theoretical foundations remain poorly understood, leading to fragmented research efforts. In this thesis, we provide an in-depth theoretical analysis and identify several key properties limiting the performance of MPNNs. Building on these findings, we propose several frameworks that address these shortcomings.

We identify and analyze two properties exhibited by many MPNNs: shared component amplification (SCA), where each message-passing iteration amplifies the same components across all feature channels, and component dominance (CD), where a single component gets increasingly amplified as more message-passing steps are applied. These properties lead to the observable phenomenon of rank collapse of node representations, which we identify as a generalization of the established over-smoothing phenomenon. By generalizing and decomposing over-smoothing, we enable a deeper understanding of MPNNs, more targeted solutions, the identification of the relevance of each property, and more precise communication within the field.

To avoid SCA, we show that utilizing multiple computational graphs or edge relations is necessary. The multi-relational split (MRS) framework transforms any existing MPNN into one that leverages multiple edge relations. We identify properties on multiple edge relations that enable the resulting MPNN to avoid SCA. Additionally, we introduce the spectral graph convolution for multiple feature channels (MIMO-GC), which naturally uses multiple computational graphs. A localized version, LMGC, serves as a framework for constructing MPNNs that approximate the MIMO-GC while inheriting its beneficial properties. We show that LMGCs can avoid SCA and achieve injectivity on multisets.

To address CD, we demonstrate a close connection between MPNNs and the PageRank algorithm. This connection allows us to transfer insights and modifications from PageRank to MPNNs. Based on personalized PageRank, we propose a variant of MPNNs that allows for infinitely many message-passing iterations, while preserving initial node features.

Collectively, these results contribute to a more comprehensive understanding of MPNNs, allowing future research to be more targeted. Our findings also establish frameworks for constructing MPNNs that exhibit favorable properties.

Acknowledgments

The past four years have shaped me profoundly — both personally and professionally — far beyond the acquisition of technical knowledge. I am deeply grateful for the time I have had: the opportunity to pursue research that truly interests me, the many exciting experiences, and to meet so many amazing people in Dortmund and on various occasions around the world.

I would like to sincerely thank Thomas Liebig, who made it possible for me to pursue my PhD. You gave me the freedom to explore topics that sparked my curiosity and supported me by allowing me to fully dedicate my time and energy to research. The opportunity to follow my interests and attend various conferences has influenced both my personal growth and academic journey.

I am also grateful to the many researchers who shared their perspectives with me and, often unknowingly, influenced my research: Francesco Di Giovanni, Konstantin Rusch, Adrián Arnaiz-Rodríguez, Pascal Esser, Nils Kriege, MoonJeong Park, Sohir Maskey, Christian Koke, and the many reviewers of my work over the years, whose feedback helped me better understand diverse viewpoints. Thank you as well to everyone I had the pleasure of interacting with at conferences, making each event uniquely memorable. A special thanks goes out to Xeniya Reich, with whom I studied countless related papers during the early phase of my PhD, laying the foundation for much of what followed. I am also thankful to Gilbert Strang and Ulrike von Luxburg for making their excellent resources on mathematical foundations publicly available — I benefited from them greatly. I am also grateful to everyone involved in our irregular monthly graph meeting: Frank Bause, Alice Moallemy-Oureh, Pascal Welke, Max Thiessen, Fabian Jogl, Sagar Malhotra, Tamara Drucks, and all others who have been involved over time.

To my co-authors — Thomas Liebig, Franka Bause, Pascal Welke, Konstantin Wüstefeld, Alice Moallemy-Oureh, Veronica Lachi, Nils Kriege, Frank Weichert, and Cedric Sanders — thank you for your collaboration. Working with you over the past four years has provided invaluable insights, improved our research outcomes, and served as a constant source of motivation.

I would like to thank my family — especially my parents, Karin and Gerd — for their unconditional support throughout all this time. A heartfelt thank you also to my brother, Stefan, who always listened with patience and shared his valuable experiences of the general research process.

Finally, I want to thank the love of my life, Hanna. Your love and support motivate me every single day. I look forward to having you by my side — now and always.

Contents

1 Introduction

1.1 Motivation

There are numerous emerging challenges across the world, from global issues such as the 17 Sustainable Development Goals outlined by the United Nations, to more specialized problems faced by individuals and organizations. Humans can address many of these challenges more effectively when suitable algorithms are available to support them. Such algorithms can assist in understanding complex phenomena, contribute to the development of viable solutions, or enable the application of solutions to recurring situations. In many domains, large volumes of data are available. This data can range from historical observations and simulations to data collected from real-world environments or digital systems. In recent years, the amount of available data has increased substantially across various tasks. Additionally, the available computational power has advanced considerably (Schmidhuber 2015). These developments underscore the importance of designing algorithms that can effectively leverage large amounts of data and computational power to address challenges of interest.

Machine learning offers a broad set of techniques well-suited to utilize available data and computational power effectively. For visual and language data, machine learning techniques have achieved substantial progress (K. He et al. 2016; Vaswani et al. 2017; Roth, Wüstefeld, et al. 2021), resulting in impactful real-world applications (OpenAI 2023; Touvron et al. 2023). These data types exhibit regular structures, with language being inherently sequential, as one word follows another. Visual data, such as images or videos, exhibits consistent spatial directionalities.

For many other challenges, data does not have such a regular structure. For example, in chemical processes, data may represent molecules, and the goal might be to identify their properties or interactions with other molecules automatically. A molecule can generally consist of any number of atoms without a clear ordering among them. The bonds between atoms likewise do not follow a regular structure. Similarly, social networks consist of individual users and their interactions, which can vary significantly across different users. Other examples of irregular data include knowledge graphs (Hogan et al. 2021) or transaction records (You et al. 2022; Fey, Hu, et al. 2024). Any such irregular structure can be represented as a graph that consists of entities and arbitrary interactions between them. Machine learning techniques for graph-structured data have not achieved the same level of success as those used in vision or language applications (Bechler-Speicher et al. 2025).

This thesis investigates the causes of the performance discrepancy of methods for graph-based machine learning compared to other machine learning methods for regular data. By theoretically analyzing the properties of graph-based machine learning methods that contribute to this discrepancy, we aim to advance our understanding of existing methods. This understanding also allows the design of targeted improvements to these methods that avoid the detrimental properties of existing methods. Based on these theoretical findings, we propose several directions to improve the effectiveness of machine learning methods for graph-structured data.

1.2 Research Questions

Our first goal is to gain a deeper understanding of machine learning methods for graph-structured data, focusing specifically on differentiable approaches known as graph neural networks (GNNs). While neural networks in general are not yet well understood, this is especially true for GNNs. GNNs commonly rely on an iterative message-passing framework where each iteration combines the representations of each node with those of its neighbors. In these models, it has been observed that increasing the number of message-passing steps often results in performance degradation. However, the specific properties of message-passing operations that cause this performance degradation, as well as the conditions under which it can be prevented, remain poorly understood. Our first research question seeks to improve our understanding of the message-passing framework and the reason behind the performance degradation:

Research Question 1. *How do established message-passing operations affect the representations, and why does repeatedly applying such operations result in a performance degradation?*

A widely considered explanation for this performance degradation is over-smoothing, a phenomenon that refers to the observation that repeatedly applying message-passing operations causes node representations to become increasingly similar to one another. Although over-smoothing has been extensively studied both empirically and theoretically, its details remain unclear. This has led to multiple, sometimes contradictory, definitions and interpretations of over-smoothing. We are interested in examining how these different perspectives emerged, and how our understanding can be clarified and improved.

Better insights into the reasons for the performance degradation are not only theoretically valuable but also practically relevant for practitioners who seek to avoid this phenomenon. More broadly, a better understanding of the message-passing process is essential for the long-term development of GNNs, as future research efforts can build on a more solid theoretical foundation.

Due to the lack of a unified explanation for the causes behind the performance degradation and over-smoothing, the proposed improvements target symptoms rather than

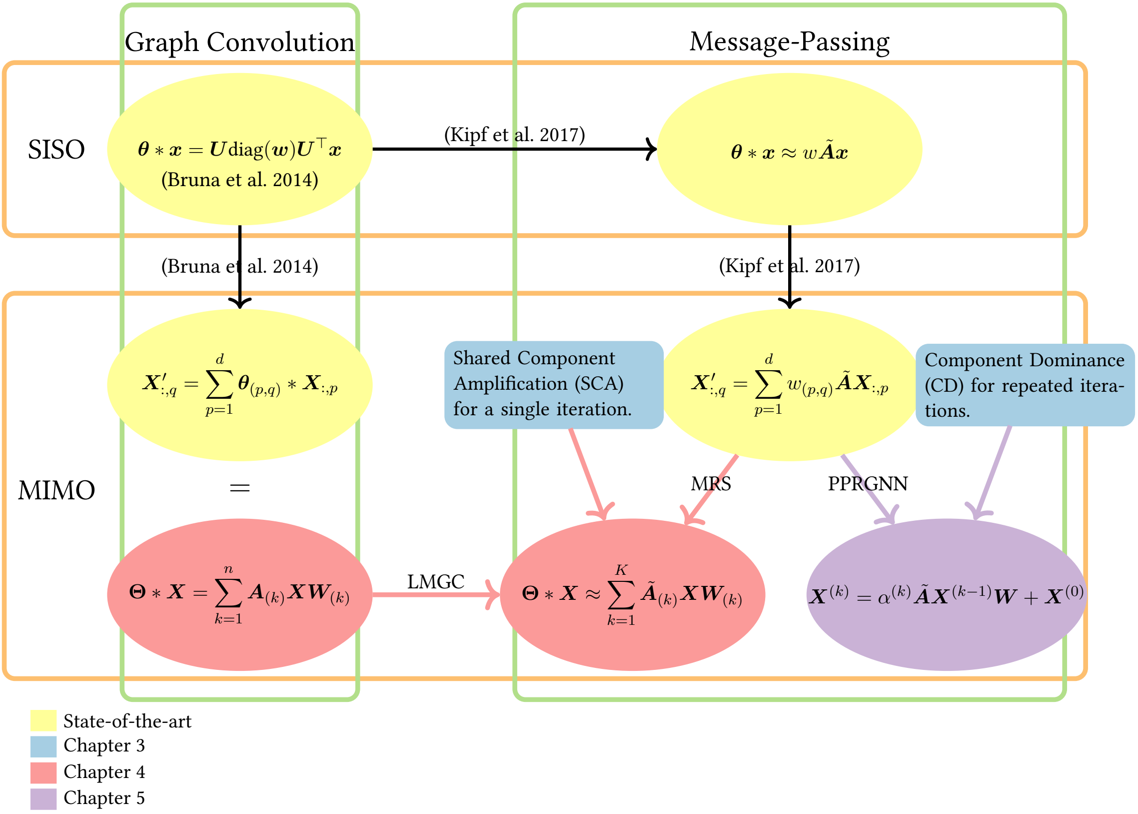


Figure 1.1: Schematic overview of this thesis and the connections between our contributions.

the currently unknown reasons behind the phenomena. The missing theoretical foundation of many recently proposed methods motivates our second research question as follows:

Research Question 2. *How can our improved understanding of message-passing operations be used to streamline the design and selection of methods to exhibit favorable properties?*

While many promising techniques have been proposed, utilizing and designing specific methods that exhibit properties we understand and demand for a challenge of interest is crucial. We aim to investigate how general frameworks can be constructed to support the design of principled methods and allow future theoretical and empirical studies within a provided framework.

1.3 Main Contributions

Following our outlined research questions, this thesis presents several key contributions to address these. We present a visual guidance through our contributions and chapters in Figure 1.1.

We conduct an in-depth study on graph convolutions and message-passing operations in GNNs, specifically examining the underlying properties that lead to the over-smoothing phenomenon (Research Question 1). As many works employ different and partially contradictory definitions, we clarify the differences in the current state-of-the-art research on over-smoothing as a first contribution (Roth and Liebig 2023). We identify that an overlapping phenomenon of vanishing norms, causing representations to approach the zero state the more message-passing iterations are applied, has led to these contrary definitions (Roth and Liebig 2023). We also provide insight into the effect of message-passing operations on node representations from a spectral perspective (Roth and Liebig 2025). We then conduct a detailed theoretical study on the effect of message-passing methods on the representations. We identify two fine-grained phenomena, namely shared component amplification (SCA) that occurs for a single message-passing step, and component dominance (CD) (Roth and Liebig 2023; Roth and Liebig 2025) that arises for the repeated application of message-passing steps. We find that the joint occurrence of these phenomena results in the general issue of rank collapse, which refers to node representations becoming increasingly close to a low-rank state. Over-smoothing occurs as a special case for particular message-passing methods. These findings extend the current theoretical understanding of GNNs and the conditions under which various phenomena arise (Roth and Liebig 2023). We also simplify our theoretical study by demonstrating that many message-passing operations can be viewed as a special case of power iteration, for which extensive theoretical insights are available (Roth 2024).

Based on our enhanced intuition and theoretical understanding of the effect of graph convolutions, we propose several frameworks for message passing that exhibit favorable properties (Research Question 2). First, we consider SCA, which we identify as always occurring for a single computational graph. We show that SCA can only be avoided when performing message-passing with multiple computational graphs, or equivalently, multigraphs (Roth and Liebig 2023; Roth, Bause, et al. 2024; Roth and Liebig 2025). We introduce two directions to obtain multiple computational graphs within the MPNN framework. The first approach starts with existing methods and outlines how multiple computational graphs can be integrated into any existing method (Roth, Bause, et al. 2024). With our proposed multi-relational splits, we decompose the computational graph of existing GNNs into multiple computational graphs by assigning each edge to one of several computational graphs. The second approach uses a novel approximation of the spectral graph convolution defined in graph Fourier space (Roth and Liebig 2025). Previous work defined the graph convolution in graph Fourier space and proposed approximations in the SISO case, and applied it to the MIMO case afterwards. We instead define the MIMO graph convolution in graph Fourier space and directly approximate it in the MIMO case. The MIMO graph convolution naturally utilizes multiple computational graphs, with each amplifying a different component in the data. Our novel approximation inherits this property and introduces a framework for message-passing that closely follows the MIMO graph convolution. We prove that both methods for obtaining multiple computational graphs solve SCA.

We relate CD as the second theoretical shortcoming related to over-smoothing in GNNs to the PageRank algorithm (Page et al. 1999) as it exhibits a phenomenon closely related to CD (Roth and Liebig 2022b). For PageRank, this shortcoming was mitigated by an adaptation called personalized PageRank (PPR), which extends the algorithm with a teleportation term that ensures a localization even as the number of iterations approaches infinity (Page et al. 1999). Inspired by this, we propose a modification of MPNNs based on PPR that similarly prevents CD even in the limit of infinitely many message-passing iterations (Roth and Liebig 2022b). We refer to our method as the personalized PageRank GNN (PPRGNN).

1.4 Outline and Covered Publications

This thesis combines and presents five individual peer-reviewed publications led by the author of this thesis. Here, we outline the structure of this thesis and refer to the publications each chapter is based on. The structure of this thesis and the connection between the chapters are also visualized in Figure 1.1.

Chapter 2: Fundamentals of Graph Machine Learning

Chapter 2 provides an overview of the fundamental concepts of graph structures and machine learning for graphs. This will be necessary to understand the details of our contributions in the following chapters. The overview includes basic graph-theoretical definitions and properties, such as the graph isomorphism problem and the graph Fourier transform (Section 2.2). We also provide details on machine learning techniques for graph-structured data (Section 2.3). There, we particularly focus on the definition of the graph convolution and message-passing neural networks.

Chapter 3: Extending Our Understanding of Graph Convolutions

In Chapter 3, we strive to find a better theoretical and intuitive understanding of graph convolutions and their localized approximations. We study the effect of message-passing operations on node representations and particularly examine the reasons behind the over-smoothing phenomenon. We provide a spectral intuition (Section 3.2.2), study the theoretical properties in-depth (Section 3.5), and then simplify the analysis for more accessible and intuitive theoretical properties (Section 3.5.4). Here, we also show that two finer-grained phenomena underlie over-smoothing: shared component amplification (SCA) and component dominance (CD). This chapter is mainly based on the following two peer-reviewed publications:

- **Andreas Roth**, Thomas Liebig (2023). "Rank Collapse Causes Over-Smoothing and Over-Correlation in Graph Neural Networks." In: Learning on Graphs Conference (LoG).
- **Andreas Roth** (2024). "Simplifying the Theory on Over-Smoothing." In: Lernen, Wissen, Daten, Analysen (LWDA).

Chapter 4: Preventing Shared Component Amplification With Multiple Computational Graphs

As one of two phenomena underlying over-smoothing, we show that message-passing operations can only prevent SCA when they utilize multiple computational graphs. We propose two methods for obtaining multiple computational graphs in this chapter. First, we demonstrate how the computational graph of existing approaches can be split into multiple computational graphs by assigning each edge to one of several relation types (Section 4.3). Second, we propose a novel way of approximating and localizing the spectral graph convolution defined in graph Fourier space (Section 4.4). We prove that both directions can effectively prevent SCA. This chapter is based on the following publications:

- **Andreas Roth**, Franka Bause, Nils M. Kriege, Thomas Liebig (2024). "Preventing Representational Rank Collapse in MPNNs by Splitting the Computational." In: Learning on Graphs Conference (LoG).
- **Andreas Roth**, Thomas Liebig (2025). "What can we learn from MIMO Graph Convolutions?" In: International Joint Conferences on Artificial Intelligence (IJCAI).

Chapter 5: Preventing Component Dominance based on Personalized PageRank

As identified in Chapter 3, the second issue underlying over-smoothing is what we refer to as component dominance. In Chapter 5, we establish the connection between this phenomenon in MPNNs and the PageRank algorithm, which exhibits a closely related phenomenon. As personalized PageRank was proposed as a solution to this issue, we show how MPNNs can adapt this solution. This chapter is based on the following peer-reviewed publication:

- **Andreas Roth**, Thomas Liebig (2022). "Transforming PageRank into an Infinite-Depth Graph Neural Network." In: Joint European conference on machine learning and knowledge discovery in databases (ECMLPKDD) (**best paper award**).

Chapter 6: Summary and Outlook

To conclude this thesis, Chapter 6 summarizes our findings, puts them into greater context, and outlines promising directions for future work.

Additional Publications

The author additionally contributed to several peer-reviewed publications that are outside the scope of this thesis. These works consider further challenges of gradient-based machine learning:

- **Andreas Roth***, Konstantin Wüstefeld*, Frank Weichert (2021). "A Data-Centric Augmentation Approach for Disturbed Sensor Image Segmentation." In: Journal of Imaging 7.10.
- **Andreas Roth**, Thomas Liebig (2022). "Forecasting Unobserved Node-States with Spatio-Temporal Graph Neural Networks." In: ICDM Workshop on Machine Learning on Graphs (MLoG).
- **Andreas Roth**, Thomas Liebig (2023). "Distilling Influences to Mitigate Prediction Churn in Graph Neural Networks." In: Asian Conference on Machine Learning (ACML).
- Cedric Sanders*, **Andreas Roth***, Thomas Liebig (2023). "Curvature-based Pooling within Graph Neural Networks." In: ECMLPKDD Workshop on Mining and Learning with Graphs (MLG).
- Veronica Lachi*, Alice Moallemy-Oureh*, **Andreas Roth***, Pascal Welke* (2025). "Expressive Pooling for Graph Neural Networks." In: Transactions on Machine Learning Research (TMLR).

* These authors contributed equally to the corresponding publication.

2 Fundamentals of Graph Machine Learning

In this chapter, we provide a brief introduction to graph machine learning and its related background topics. We establish our notation and provide necessary background knowledge underlying the following chapters. These fundamentals will help to understand the current state-of-the-art, our considered challenges, and our contributions. We will first introduce required graph theoretical basics in Section 2.2, including relevant spectral properties, particularly the concept of the graph Fourier transform. Building on this, we will introduce the topic of machine learning for graph-structured data, with a focus on differentiable approaches and the convolution on graphs in Section 2.3. Based on this, we demonstrate how message-passing neural networks (MPNNs) are derived from graph convolution and how machine learning tasks based on graph-structured data are typically addressed using these methods.

2.1 Notation

The notation used in this thesis closely follows that of Goodfellow et al. (2016). A set is denoted by a calligraphic symbol, such as $\mathcal{A} = \{ a_1, a_2, \dots \}$, with special notation reserved for the set of natural numbers $\mathbb{N}$ and the set of real numbers $\mathbb{R}$. For $n \in \mathbb{N}$, we use $[n] = \{ 1, \dots, n \}$ to denote the set of natural numbers from 1 to n. A multiset is written as $\mathcal{A} = \{\{ a_1, \dots, a_1, a_2, \dots \}\}$, where elements may appear multiple times. Vectors are denoted in bold lowercase, e.g., $\boldsymbol{a}$, with the i-th element written as a_i. Matrices are denoted in bold uppercase, e.g., $\boldsymbol{A}$, with the i-th row written as $\boldsymbol{A}_{i,:}$, the j-th column as $\boldsymbol{A}_{:,j}$, and the element in the i-th row and j-th column as $A_{i,j}$. Further notation that is specific to individual sections or methods is introduced as needed.

2.2 Graph Theory

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined as a finite set of nodes $\mathcal{V} = \{v_1, \dots, v_n\}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ that represents connections between pairs of nodes. An attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, x)$ is a graph extended by a function $x \colon \mathcal{V} \to \mathbb{R}^d$ that assigns a d-dimensional vector $x(v_i) \in \mathbb{R}^d$ of real numbers as attributes to every node $v_i \in \mathcal{V}$. Unless otherwise indicated, we assume graphs to be undirected, i.e., $(v_i, v_j) \in \mathcal{E}$ whenever $(v_j, v_i) \in \mathcal{E}$. Graphs that do not satisfy this condition are called directed graphs. For every $(v_i, v_j) \in \mathcal{E}$, we refer to v_i and v_j as neighboring or adjacent nodes. The set of nodes adjacent to node $v_i \in \mathcal{V}$ is represented by $\mathcal{N}_i = \{ v_j \mid (v_j, v_i) \in \mathcal{E} \}$. The degree of node v_i is defined as the number of neighboring nodes $d_i = |\mathcal{N}_i|$, where $| \cdot |$ indicates the number of elements in the set. A graph is bipartite when there exist $\mathcal{V}_1, \mathcal{V}_2 \subset \mathcal{V}$ with $\mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$ and $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$ such that for every edge $(v_i, v_j) \in \mathcal{E}$ the two corresponding nodes v_i and v_j are in a different subset, i.e., we have either $v_i \in \mathcal{V}_1$ and $v_j \in \mathcal{V}_2$ or $v_i \in \mathcal{V}_2$ and $v_j \in \mathcal{V}_1$. Due to the properties we will introduce in Section 2.2.2, we assume graphs to be non-bipartite.

Graphs are typically processed by their corresponding matrix representations for simpler notation and hardware acceleration. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ can be equivalently described by its adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n \times n}$ where $A_{i,j} = 1$ if $(v_i, v_j) \in \mathcal{E}$ and $A_{i,j} = 0$ otherwise. The degree matrix $\boldsymbol{D} \in \mathbb{N}^{n \times n}$ is a diagonal matrix that contains the node degrees $D_{i,i} = d_i$ on its diagonal. For attributed graphs, the function $x \colon \mathcal{V} \to \mathbb{R}^d$ is represented as an equivalent matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ where every row $\boldsymbol{X}_{i,:} = x(v_i)$ corresponds to the attributes of the corresponding node v_i. Analogously to the set notation, we refer to $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ as a graph in matrix notation.

2.2.1 Graph Isomorphism

A key challenge of graph theory is determining whether two graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ are isomorphic (Luks 1982). Two graphs are said to be isomorphic when there exists a bijective function $\phi \colon \mathcal{V}_1 \to \mathcal{V}_2$ mapping each node of graph $\mathcal{G}_1$ to exactly one node of graph $\mathcal{G}_2$ such that $(\phi(v_i), \phi(v_j)) \in \mathcal{E}_2$ if and only if $(v_i, v_j) \in \mathcal{E}_1$ for every $v_i, v_j \in \mathcal{V}_1$. We denote two isomorphic graphs as $\mathcal{G}_1 \simeq \mathcal{G}_2$. For attributed graphs, we additionally require $x_1(v_i) = x_2(\phi(v_i))$ for all $v_i \in \mathcal{V}_1$. There are currently no polynomial-time algorithms known for this task (Babai 2016).

When representing graphs in their matrix notation $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$, the ordering of the nodes is arbitrary. Thus, two isomorphic graphs $\mathcal{G}_1 = (\boldsymbol{A}_1, \boldsymbol{X}_1)$ and $\mathcal{G} = (\boldsymbol{A}_2, \boldsymbol{X}_2)$ may have a different ordering of rows and columns in $\boldsymbol{A}$ and $\boldsymbol{X}$. Deciding whether two graphs with corresponding adjacency matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ are isomorphic is equivalent to determining whether there exists a permutation matrix $\boldsymbol{P}$ such that $\boldsymbol{P}\boldsymbol{A}_1\boldsymbol{P}^\top = \boldsymbol{A}_2$. A permutation matrix $\boldsymbol{P}$ contains a single one in each row and column, with all other entries being zero. Applying such a permutation matrix to an adjacency matrix corresponds to modifying the ordering of the nodes. For matrix representations $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ of node attributes, we additionally require the permutation matrix $\boldsymbol{P}$ to satisfy $\boldsymbol{P}\boldsymbol{X}_1 = \boldsymbol{X}_2$. A function that maps graphs $\mathcal{G}$ to a vector space and is invariant under graph isomorphisms can be equivalently expressed as a function over matrix representations of the graphs that is invariant under permutations of node orderings. Otherwise, such a function f would assign different outputs to isomorphic graphs. We define this property as follows:

Definition 2.1 (Permutation Invariance). *A function f is* permutation invariant *under node orderings if*

$$f(\boldsymbol{A}, \boldsymbol{X}) = f\left(\boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^\top, \boldsymbol{P}\boldsymbol{X}\right) \tag{2.1}$$

for all $n \in \mathbb{N}$, permutation matrices $\boldsymbol{P} \in \{0,1\}^{n\times n}$, adjacency matrices $\boldsymbol{A} \in \{0,1\}^{n\times n}$, and feature matrices $\boldsymbol{X} \in \mathbb{R}^{n\times d}$.

The methods considered in this thesis are typically stated to operate on matrix representations of graphs. All of these methods are permutation invariant and can be equivalently reformulated to operate directly on the original graph representation.

2.2.2 Spectral Graph Theory

Key to the methods considered in this work is the field of spectral graph theory (Chung 1997; Luxburg 2007). Based on the matrix notation of a graph, spectral graph theory involves properties of different types of graph Laplacians and their spectra. We will describe graph Laplacians and their necessary properties for this work in the following.

Graph Laplacians We now introduce graph Laplacians and some of their properties. This part is mainly based on Luxburg (2007). Consider an undirected and non-bipartite graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ of n nodes in matrix notation with adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n\times n}$ and degree matrix $\boldsymbol{D} \in \mathbb{N}^{n\times n}$. The unnormalized graph Laplacian is defined as the matrix

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}\,. \tag{2.2}$$

When applied to a vector $\boldsymbol{x} \in \mathbb{R}^n$, each value of $\boldsymbol{L}\boldsymbol{x}$ represents the sum of the differences between the state of a node and all of its neighbors, i.e.,

$$(\boldsymbol{L}\boldsymbol{x})_i = \sum_{v_j \in \mathcal{N}_i} x_i - x_j\,. \tag{2.3}$$

We will also rely on two normalized graph Laplacians. For this, we utilize two normalized adjacency matrices. The symmetrically normalized adjacency matrix $\boldsymbol{A}_{\text{sym}} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ scales every entry $A_{i,j} = \frac{1}{\sqrt{d_i \cdot d_j}}$ by the degrees of the corresponding nodes. The random walk normalized adjacency matrix $\boldsymbol{A}_{\text{rw}} = \boldsymbol{D}^{-1}\boldsymbol{A}$ normalizes the values of each row to sum to one, i.e., we have $A_{i,j} = \frac{1}{d_i}$. The corresponding symmetrically normalized graph Laplacian is defined as

$$\boldsymbol{L}_{\text{sym}} = \boldsymbol{I} - \boldsymbol{A}_{\text{sym}}\,, \tag{2.4}$$

and the random walk normalized graph Laplacian as

$$\boldsymbol{L}_{\text{rw}} = \boldsymbol{I} - \boldsymbol{A}_{\text{rw}}\,. \tag{2.5}$$

Similarly to Equation (2.3), applying these to a vector $\boldsymbol{x}$ results in the weighted difference between the state of a node and its neighboring states. These graph Laplacians have several key properties that we will rely on throughout this thesis. All three graph Laplacians $\boldsymbol{L}, \boldsymbol{L}_{\text{sym}}$, and $\boldsymbol{L}_{\text{rw}}$ are positive semi-definite matrices. Given a vector $x \in \mathbb{R}^n$, the Dirichlet sum or Dirichlet energy

$$\boldsymbol{x}^\top \boldsymbol{L}\boldsymbol{x} = \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{N}_i} (x_i - x_j)^2 \tag{2.6}$$

describes the sum of squared differences in values between adjacent nodes (Luxburg 2007). The Dirichlet energy for the normalized graph Laplacians is defined equivalently. As $\boldsymbol{L}$ and $\boldsymbol{L}_{\text{sym}}$ are symmetric matrices, their eigenvectors are orthogonal.

We denote the i-th largest eigenvalue of a matrix $\boldsymbol{P}$ as $\lambda_i^{(\boldsymbol{P})}$. Utilizing the assumed undirected property, the smallest eigenvalues $\lambda_n^{(\boldsymbol{L})} = \lambda_n^{(\boldsymbol{L}_{\text{sym}})} = \lambda_n^{(\boldsymbol{L}_{\text{rw}})} = 0$ of the graph Laplacians $\boldsymbol{L}, \boldsymbol{L}_{\text{sym}}, \boldsymbol{L}_{\text{rw}}$ are zero with associated eigenvector $\mathbf{1}$ for $\boldsymbol{L}$ and $\boldsymbol{L}_{\text{rw}}$, and $\boldsymbol{D}^{1/2}\mathbf{1}$ for $\boldsymbol{L}_{\text{sym}}$ (Luxburg 2007).

Given the non-bipartite property, we further have $\lambda_i \in [0, 2)$ for $\boldsymbol{L}_{\text{sym}}$ and $\boldsymbol{L}_{\text{rw}}$ and all $i \in [n]$. The random walk normalized graph Laplacian $\boldsymbol{L}_{\text{rw}}$ and the corresponding normalized adjacency matrix $\boldsymbol{A}_{\text{rw}}$ have the same eigenvectors with the associated eigenvalues shifted as $\lambda_{n-i+1}^{(\boldsymbol{A}_{\text{rw}})} = 1 - \lambda_i^{(\boldsymbol{L}_{\text{rw}})}$. This is equivalent for the symmetrically normalized graph Laplacian $\boldsymbol{L}_{\text{sym}}$ and the symmetrically normalized adjacency matrix $\boldsymbol{A}_{\text{sym}}$. These properties are further exploited to construct the graph Fourier transform, which we will introduce next.

Graph Fourier Transform The Fourier transform F maps a given function f defined on $\mathbb{R}$ into another function $g = F(f)$ defined on $\mathbb{R}$. The function g is typically interpreted as the frequency domain representation of f. The Fourier transform exploits the periodicity and regularity of $\mathbb{R}$ by decomposing f into sinusoidal components with varying frequencies. Evaluating the Fourier transformed function $g(a)$ at some frequency a represents the magnitude of this frequency for the function f. In the discrete case, the discrete Fourier transform is defined over finite sequences that are equally spaced.

For graphs, the aim is to find a frequency domain representation of a graph signal $x\colon \mathcal{V} \to \mathbb{R}$, also referred to as node attributes. However, as graphs do not exhibit any regularity in general, the discrete Fourier transform cannot be applied to x. Instead, the graph Fourier transform

$$F(\boldsymbol{x}) = \boldsymbol{U}^\top \boldsymbol{x} \tag{2.7}$$

is defined as the projection of the vectorized node attributes $\boldsymbol{x} \in \mathbb{R}^n$ onto the orthogonal eigenvectors $\boldsymbol{U} \in \mathbb{R}^{n \times n}$ of $\boldsymbol{L}_{\text{sym}}$ (Sandryhaila et al. 2013; Shuman et al. 2013). The eigenvectors $\boldsymbol{U}$ of the graph Laplacian $\boldsymbol{L}_{\text{sym}}$ are used as the graph Fourier base for the graph Fourier transform, as they exhibit similar properties to sinusoidal components. The corresponding eigenvalues $\boldsymbol{\Lambda}_{\text{sym}}$ of $\boldsymbol{L}_{\text{sym}}$ represent the frequencies. The eigenvector corresponding to the smallest eigenvalue or frequency λ_1 is $\boldsymbol{U}_{:,1} = \boldsymbol{D}^{1/2}\mathbf{1}$ for $\boldsymbol{L}_{\text{sym}}$, i.e., it has similar values for all nodes. For larger frequencies λ_i, the corresponding eigenvectors

$\boldsymbol{U}_{:,i}$ increasingly oscillate between adjacent nodes. The eigenvector associated with the second smallest eigenvalue, also referred to as the Fiedler vector (Fiedler 1973), is used for graph cutting due to $(U_{\text{sym}})_{i,2}$ being > 0 for all nodes in one of the partitions and < 0 for all nodes in the other partition (Fiedler 1973). The eigenvectors associated with higher frequencies increasingly oscillate across adjacent nodes (Sandryhaila et al. 2013; Shuman et al. 2013). Assuming $\boldsymbol{U}$ is orthonormal, the inverse graph Fourier transform

$$F^{-1}(F(\boldsymbol{x})) = \boldsymbol{U}F(\boldsymbol{x}) = \boldsymbol{U}\boldsymbol{U}^{\top}\boldsymbol{x} = \boldsymbol{x} \tag{2.8}$$

is performed by multiplying a frequency domain representation with the graph Fourier basis $\boldsymbol{U}$.

The fast Fourier transform reduces the runtime of the discrete Fourier transform to $\mathcal{O}(n \log n)$ for sequences of length n. However, as it exploits the regularity of the space, the same technique cannot be applied to the graph Fourier transform. There is currently no subquadratic algorithm known to compute the graph Fourier transform. As the graph Fourier basis $\boldsymbol{U}$ depends on the structure of a given graph, it also cannot be applied across distinct graphs.

2.3 Graph Machine Learning

In machine learning, we are generally interested in making predictions based on previously observed data. While the field of machine learning encompasses a multitude of different tasks and methods, we here focus on the foundations of machine learning for graph-structured data. We describe the task of classifying graphs, motivate the use of the convolution operation on graphs, and demonstrate how it can be efficiently approximated by message-passing neural networks.

2.3.1 Graph Classification

Classifying attributed graphs into a set of classes $\mathcal{Y}$ is similar to classification tasks for other domains. The key is to find functions that map from a set of graphs to the set of classes $\mathcal{Y}$. We start by defining such functions as graph classifiers:

Definition 2.2 (Graph Classifier). *Let $\mathcal{H}$ denote a set of attributed graphs, and let $\mathcal{Y}$ be a finite set of discrete class labels. A* graph classifier *is a function $f\colon \mathcal{H} \to \mathcal{Y}$ that assigns a label $f(\mathcal{G}) \in \mathcal{Y}$ to each graph $\mathcal{G} \in \mathcal{H}$.*

As described in Definition 2.1, a graph classifier f is typically defined as a function on the matrix form of an attributed graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ that is invariant under node permutations.

The performance of a given graph classifier f is computed using a suitable metric, e.g., accuracy or the F1-score. This is typically achieved in the supervised learning setting, where a dataset $\mathcal{D} = \{(\mathcal{G}_i, y_i) \mid i \in [m]\}$ of m graphs for which the label is known is given. The dataset can be split into training, validation, and testing graphs, and the classifier can be optimized over a family of classifiers using the training data. The validation graphs are utilized to ensure good generalization to the underlying distribution of graphs, and the testing graphs are used to approximate the performance of a selected graph classifier over the true distribution.

Challenges in Graph Machine Learning The particular challenge of graph classification is that well-performing graph classifiers typically need to consider both the structure of the data and the data itself. For other domains, all inputs to a classifier typically have a shared structure, e.g., vectors with the same number of features, text with a sequential structure, or images with a regular grid structure. There are no such regularities for graphs. Each graph can have a different number of nodes and any connectivity between these nodes. Thus, each node may have a different number of neighboring nodes, and there is no inherent ordering or directionality for neighboring nodes. Such differences in the data structure are key to many graph classification tasks. For example, in molecular datasets, a graph may correspond to a protein, where nodes represent the atoms and edges represent the bonds between them. The behavior of proteins changes drastically when their atoms or bonds are modified (Pérez-Villanueva et al. 2015). As such, the data structure of a graph is a key feature.

Expressivity Ideally, considered graph classifiers can distinguish all graphs from the considered set of graphs. This would allow us to find a graph classifier that assigns all graphs to their correct class. Based on the graph isomorphism problem (Section 2.2.1), any graph classifier f with polynomial runtime cannot distinguish some graphs $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{H}$ and must map them to the same label $y \in \mathcal{Y}$. The expressivity of graph classifiers typically describes their ability to distinguish non-isomorphic graphs.

An established method for testing graph isomorphism is the Weisfeiler-Leman (WL) test (Weisfeiler et al. 1968). Given a set of colors $\mathcal{C}$, the WL test initially assigns the same color $c_i^{(0)}$ to every node $v_i \in \mathcal{V}$. The test then iteratively refines the color of each node $v_i \in \mathcal{V}$ using the update rule

$$c_i^{(k)} = \phi\left(c_i^{(k-1)}, \left\{\left\{\, c_j^{(k-1)} \mid v_j \in \mathcal{N}_i \,\right\}\right\}\right) , \tag{2.9}$$

where ϕ is an injective function and $c_i^{(k-1)}$ is the color of node $v_i \in \mathcal{V}$ at iteration $k-1$. The new color of each node is unique for different multisets of neighboring colors or previous colors. Two graphs are non-isomorphic if the multisets of node colors differ at any iteration i. However, when the multisets of colors for two graphs remain identical across all iterations, it cannot be concluded that these are isomorphic. Despite this limitation, it is computationally efficient and can distinguish a large class of non-isomorphic graphs. Graph classifiers are often compared to the WL test in terms of their expressivity (Shervashidze, Schweitzer, et al. 2011; Xu, Hu, et al. 2019; Morris, Ritzert, et al. 2019; Kriege, Johansson, et al. 2020) and methods are constructed to improve the expressivity (Bouritsas et al. 2023; Sanders et al. 2023; Lachi et al. 2025).

Graph Kernels Graph kernels are a classical and effective approach to graph classification (Gärtner et al. 2003; Kashima et al. 2003; Shervashidze, Schweitzer, et al. 2011; Kriege, Johansson, et al. 2020). A graph kernel defines a similarity measure between pairs of graphs. To classify a graph, similarity scores are computed with respect to all training graphs and used as feature vectors for a vector-based classifier that operates on vector inputs. A common approach is to combine graph kernels with support vector machines (SVMs) (Cortes et al. 1995), allowing for the classification of graphs based on these feature vectors. Different graph kernels vary in their definition of similarity, and most can be grouped into the following four design paradigms (Kriege, Johansson, et al. 2020). The first paradigm is referred to as neighborhood aggregation-based graph kernels. These assign labels to nodes based on their local neighborhood. Similar labels indicate a high similarity between those nodes. The Weisfeiler-Leman kernel (Shervashidze, Schweitzer, et al. 2011) and its modifications (Hido et al. 2009; Morris, Kersting, et al. 2017) are prominent examples. The second paradigm defines assignments between the nodes in two graphs (Fröhlich et al. 2005; Johansson and Dubhashi 2015; Woznica et al. 2010; Kriege, Giscard, and Wilson 2016). Another paradigm considers sets of subgraphs and measures the similarity of two graphs by comparing these sets (Horváth et al. 2004; Shervashidze, Vishwanathan, et al. 2009; Costa et al. 2010). As the last paradigm, graph kernels based on walks and paths were proposed. These include shortest-path kernels (Borgwardt et al. 2005; Hermansson et al. 2015) and random walk kernels (Gärtner et al. 2003; Kashima et al. 2003; Vishwanathan et al. 2010; Kriege, Neumann, Kersting, et al. 2014; Kriege, Neumann, Morris, et al. 2019). Extensions of these kernels address challenges such as handling continuous node attributes (Kriege and Mutzel 2012; Feragen et al. 2013; Orsini et al. 2015; Morris, Kriege, Kersting, et al. 2016), improving computational efficiency (Kriege, Giscard, Bause, et al. 2019), and analyzing the theoretical expressivity of kernel-based graph classifiers (Gärtner et al. 2003; Johansson, Jethava, et al. 2014; Johansson and Dubhashi 2015; Oneto et al. 2017; Kriege, Morris, et al. 2018), as discussed in Section 2.3.1.

Despite their strong performance on benchmark datasets (Cai et al. 2018; Kriege, Johansson, et al. 2020), this thesis focuses on differentiable methods, which have shown better scalability and performance for domains for which large amounts of data are available (K. He et al. 2016; Vaswani et al. 2017).

2.3.2 Differentiable Classifiers

Due to the improved availability of larger amounts of data and computing resources (Hu et al. 2021), graph classifiers that can effectively benefit from these developments are desired. Differentiable classifiers have emerged as a promising method across various domains, which can benefit more effectively from large amounts of data and computing resources (LeCun, Bengio, et al. 2015). Here, graph classifiers produce a continuous output, e.g., class probabilities $\hat{\boldsymbol{y}} \in [0,1]^{|\mathcal{Y}|}$, from which a single class can be predicted using the class with maximal probability. For notational simplicity, we will further define graph classifiers to take the matrix notation of a graph as input directly, i.e., it is a function $f(\boldsymbol{A}, \boldsymbol{X}) = \hat{\boldsymbol{y}}$ taking the adjacency matrix $\boldsymbol{A}$ and a matrix of node representations $\boldsymbol{X}$ as arguments.

Based on the continuous class probabilities $\hat{\boldsymbol{y}} = f(\boldsymbol{A}, \boldsymbol{X})$ and the true label $y \in [|\mathcal{Y}|]$ in numerical form, we compute a differentiable loss $\ell\left(y, \hat{\boldsymbol{y}}\right)$, where ℓ for graph classification is typically chosen to be the cross-entropy loss

$$\ell\left(y, \hat{\boldsymbol{y}}\right) = - \sum_{c \in [|\mathcal{Y}|]} \mathbb{1}\left(c = y\right) \log\left(\hat{y}_c\right) \,, \tag{2.10}$$

where $\mathbb{1}(\cdot)$ is the indicator function that is one when the condition provided as its argument is true, and zero otherwise. A differentiable classifier contains a set of parameters $\mathcal{W}$, for which the partial derivative $\frac{\partial \ell(y, \hat{\boldsymbol{y}})}{\partial w}$ can be computed for each parameter $w \in \mathcal{W}$. A differentiable classifier typically consists of a sequence of differentiable operations, so that the partial derivative can be decomposed into a sequence of partial derivatives of each operation under the use of the chain rule. Gradient descent-based optimization approaches then update each parameter w as

$$w' = w - \lambda \frac{\partial \ell(y, \hat{\boldsymbol{y}})}{\partial w} \tag{2.11}$$

where $\lambda \in \mathbb{R}$ is called the learning rate. These updated parameters $\mathcal{W}'$ are then used within a new graph classifier, and the process is iterated for different graphs. Updating the parameters once is referred to as an optimization step. An epoch refers to iterating over all graphs in the dataset once. The full training process typically consists of multiple epochs. While this optimization procedure uses a single graph per optimization step, variations like batch gradient descent use the average partial derivative over multiple graphs for each optimization step. This end-to-end optimization or learning has been shown to scale effectively to large amounts of available data in other domains, such as computer vision (K. He et al. 2016) or natural language processing (Vaswani et al. 2017).

A differentiable graph classifier is typically split into two parts. The final part is a function

$$\omega(\boldsymbol{X}) = \hat{\boldsymbol{y}} \tag{2.12}$$

that typically takes a matrix of node representations $\boldsymbol{X}$ as the sole argument of a permutation invariant function ω mapping $\boldsymbol{X}$ to class probabilities $\hat{\boldsymbol{y}} \in [0,1]^{|\mathcal{Y}|}$. This permutation invariant function is equivalent to operating on multisets $\{\{\, x(v_i) \mid v_i \in \mathcal{V} \,\}\}$ for a graph signal x in function form. For example, all rows of $\boldsymbol{X}$ may be summed into a single vector. Then, a differentiable function taking a vector as an argument, such as a multi-layer perceptron (MLP) (Amari 1967), can be applied.

As a classifier ω operates only on the node representations $\boldsymbol{X}$ and not on the structural representation $\boldsymbol{A}$ of the graph, the node representations $\boldsymbol{X}$ need to capture structural properties of the graph and information about interactions between node attributes. Initially available node representations $\boldsymbol{X}$ may not capture these properties. Thus, considerable research has been dedicated to obtaining more informative node representations $\boldsymbol{X}'$ that capture the structural information and interactions between node representations of a given attributed graph. This can be achieved using the graph convolution and its approximations, which we will introduce next.

Convolution on Graphs

The convolution is a classic operation in signal processing and probability theory. In machine learning, it has been particularly successful as a part of differentiable classifiers in computer vision. It is an operation on two functions that produces a third function by shifting one of the functions. In the graph domain, the two input functions typically correspond to a graph signal $x : \mathcal{V} \rightarrow \mathbb{R}$ and a filter $\theta : \mathcal{V} \rightarrow \mathbb{R}$ for nodes $\mathcal{V}$ of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, x)$. The graph convolution $*$ produces a function $\theta * x \colon \mathcal{V} \rightarrow \mathbb{R}$. For notational simplicity, we will use the equivalent matrix notation $\boldsymbol{x}' = \boldsymbol{\theta} * \boldsymbol{x}$ where $\boldsymbol{x}', \boldsymbol{x}, \boldsymbol{\theta} \in \mathbb{R}^{|\mathcal{V}|}$ for the graph convolution. While the convolution is typically defined using the shift operation of a given domain, there is no shift operation available for graphs, and the convolution does not directly apply to graphs. Instead, the graph convolution is defined based on the convolution theorem (Oppenheim et al. 2013). It states that the Fourier transform of the convolution $*$ of two functions is equal to the element-wise product of the Fourier transformed functions. However, the Fourier transform is also not applicable to graphs, as detailed in Section 2.2.2. Instead, the Fourier transform is defined as the graph Fourier transform and the graph convolution in graph Fourier space accordingly as

$$F(\boldsymbol{\theta} * \boldsymbol{x}) = F(\boldsymbol{\theta}) \odot F(\boldsymbol{x}) \,, \tag{2.13}$$

where F is the graph Fourier transform (Hammond et al. 2011; Bruna et al. 2014).

As described in Section 2.2.2, the graph Fourier transform $F(\boldsymbol{x}) = \boldsymbol{U}^\top \boldsymbol{x}$ is defined as the projection of $\boldsymbol{x}$ onto the eigenvectors $\boldsymbol{U}$ of the graph Laplacian $\boldsymbol{L}_{\text{sym}}$ of $\mathcal{G}$ (Bruna et al. 2014). Using the inverse graph Fourier transform $F^{-1}(\boldsymbol{x}) = \boldsymbol{U}\boldsymbol{x}$ (see Section 2.2.2), the graph convolution is computed as

$$\begin{aligned} \boldsymbol{\theta} * \boldsymbol{x} &= \boldsymbol{U} \left(\boldsymbol{U}^\top \boldsymbol{\theta} \odot \boldsymbol{U}^\top \boldsymbol{x} \right) \\ &= \boldsymbol{U} \operatorname{diag}(\boldsymbol{w}) \boldsymbol{U}^\top \boldsymbol{x} \end{aligned} \tag{2.14}$$

where $\boldsymbol{w} = \boldsymbol{U}^\top \boldsymbol{\theta}$ is the vector containing the filter coefficients in graph Fourier space. Due to this definition utilizing spectral properties of the graph Laplacian, it is also called the spectral graph convolution (Bruna et al. 2014; Henaff et al. 2015). The graph convolution is differentiable, i.e., the gradient $\frac{\partial(\boldsymbol{\theta} * \boldsymbol{x})}{\partial \boldsymbol{\theta}}$ with respect to the filter $\boldsymbol{\theta}$ can be directly computed.

In Section 2.2.1, we discussed permutation invariance as a property for functions operating on matrices of graphs. However, while we considered there functions mapping from the graph domain to the vector domain, here we consider functions mapping from the graph domain to the graph domain. As such, the arbitrary ordering of the vectors $\boldsymbol{x}$ and $\boldsymbol{x}' = \boldsymbol{\theta} * \boldsymbol{x}$ should align. This property of a function is called permutation equivariance. Similar to the definition of permutation invariance in Definition 2.1, permutation equivariance is generally defined as the existence of a permutation matrix that permutes the input and output in the same way:

Definition 2.3 (Permutation Equivariance). *A function f is* permutation equivariant *under node orderings if*

$$\boldsymbol{P} f(\boldsymbol{A}, \boldsymbol{X}) = f\left(\boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^\top, \boldsymbol{P}\boldsymbol{X}\right) . \tag{2.15}$$

for all $n \in \mathbb{N}$ and a $d \in \mathbb{N}$, permutation matrices $\boldsymbol{P} \in \{0,1\}^{n \times n}$, adjacency matrices $\boldsymbol{A} \in \{0,1\}^{n \times n}$, and node representation matrices $\boldsymbol{X} \in \mathbb{R}^{n \times d}$.

The graph convolution is a permutation-equivariant function

$$\psi(\boldsymbol{A}, \boldsymbol{x}) = \boldsymbol{\theta} * \boldsymbol{x} . \tag{2.16}$$

The output $\boldsymbol{x}' = \psi(\boldsymbol{A}, \boldsymbol{x})$ of a graph convolution can be viewed as a new graph signal or vector of node representations that captures structural information about the graph and interactions between node attributes. These new node representations $\boldsymbol{x}'$ can then be used as the argument for further graph convolutions or a differentiable graph classifier ω, as given by Equation (2.12). It is also end-to-end differentiable, allowing the computation of the gradients $\frac{\partial \omega(\psi(\boldsymbol{A},\boldsymbol{x}))}{\partial \boldsymbol{\theta}}$ and optimization by some form of gradient descent.

However, this convolution is typically not applied for multiple reasons. Most importantly, the graph Fourier transform F depends on the graph $\mathcal{G}$. The graph Fourier basis vectors and the number of such vectors are typically distinct across graphs. Therefore, the filter $\boldsymbol{\theta}$ is not applicable across distinct graphs. Another disadvantage of this graph convolution is the necessity to compute the eigendecomposition for each graph to obtain the graph Fourier transform, which has quadratic complexity. As $\boldsymbol{U}$ is a dense matrix, applying the graph Fourier transform F also has quadratic runtime complexity. Thus, approximations of the spectral graph convolution are typically considered.

Message-Passing Neural Networks

To reduce the computational complexity of the graph convolution, the message-passing scheme was derived as a localized approximation of the graph convolution (Kipf et al. 2017). It is based on the idea of updating the representations of each node by incorporating the node representations of all adjacent nodes. Given a graph with adjacency matrix $\boldsymbol{A}$ and node representations $\boldsymbol{X}$, the message-passing scheme updates the node representations $\boldsymbol{X}' = g(\boldsymbol{A}, \boldsymbol{X})$ by combining the representation of each node with all neighboring node representations. It is generally defined as a node-wise function

$$(g(\boldsymbol{A}, \boldsymbol{X}))_{i,:} = \phi\left(\boldsymbol{X}_{i,:}, \bigoplus \left\{\!\!\left\{\, \alpha_{(i,j)} \cdot \psi\left(\boldsymbol{X}_{i,:}, \boldsymbol{X}_{j,:}\right) \mid v_j \in \mathcal{N}_i \,\right\}\!\!\right\}\right) \tag{2.17}$$

where ψ and ϕ are functions on pairs of node features, $\alpha_{(i,j)} \in \mathbb{R}$ is an edge-specific scalar, and $\oplus$ is a permutation invariant set aggregation function, e.g., sum or mean. Message-passing operations are localized operations due to the aggregation over the set of neighboring nodes $\mathcal{N}_i$, rather than over all nodes, as in the spectral graph convolution. It can also be applied across different graphs (Hamilton et al. 2017). By this definition, any such message-passing function g is permutation equivariant. Similar to the graph convolution, the output $\boldsymbol{X}' = g(\boldsymbol{A}, \boldsymbol{X})$ of Equation (2.17) represents an updated node representation matrix that is supposed to contain structural properties and information about interactions of node representations.

Examples To provide intuition for the message-passing scheme, we now present three established examples. We assume the node representation matrices have shapes $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{X}' \in \mathbb{R}^{n \times d'}$. First, the graph convolutional network (GCN) (Kipf et al. 2017) was proposed as a localized approximation of the spectral graph convolution that updates node representations as

$$\boldsymbol{X}'_{i,:} = \sum_{v_j \in \mathcal{N}_i} \frac{1}{\sqrt{d_i d_j}} \boldsymbol{X}_{j,:} \boldsymbol{W} \tag{2.18}$$

where $\boldsymbol{X}_{j,:} \in \mathbb{R}^{1 \times d}$ and $\boldsymbol{W} \in \mathbb{R}^{d \times d'}$ is a feature transformation.

The SAGE convolution from GraphSAGE (Hamilton et al. 2017) utilizes a second feature transformation to control the effect of the previous node representations on the resulting ones. This leads to the updated node representations

$$\boldsymbol{X}'_{i,:} = \boldsymbol{X}_{i,:} \boldsymbol{W}_{(1)} + \sum_{v_j \in \mathcal{N}_i} \frac{1}{d_i} \boldsymbol{X}_{j,:} \boldsymbol{W}_{(2)} \,, \tag{2.19}$$

where $\boldsymbol{W}_{(1)}, \boldsymbol{W}_{(2)} \in \mathbb{R}^{d \times d'}$ are two feature transformation matrices of the same shape.

The graph attention network (GAT) (Velickovic et al. 2018) learns edge weights as a function of the adjacent node representations. The updated node representations are then obtained as

$$\boldsymbol{X}'_{i,:} = \sum_{v_j \in \mathcal{N}_i} a_{(i,j)} \boldsymbol{X}_{j,:} \boldsymbol{W} \tag{2.20}$$

where $a_{(i,j)}$ are the attention coefficients that satisfy $a_{(i,j)} > 0$ for every edge (v_i, v_j) and $\sum_{v_j \in \mathcal{N}_i} a_{(i,j)} = 1$ for all $i \in [n]$. GAT is often employed with multiple attention heads, which refers to computing multiple attention-based updates with independent parameters and aggregating the results.

Message-Passing Neural Networks for Graph Classification As a single message-passing step can only produce node representations that are affected by the direct neighbors of each node, these are typically applied iteratively. Such a sequence

$$\rho\left(\boldsymbol{A}, \boldsymbol{X}\right) := g_k\left(\boldsymbol{A}, g_{k-1}\left(\ldots g_1\left(\boldsymbol{A}, \boldsymbol{X}\right)\ldots\right)\right) \tag{2.21}$$

of k message-passing iterations $g_1, \ldots, g_k$ produces the final node representations. A message-passing neural network (MPNN) is then defined as a graph classifier

$$f(\boldsymbol{A}, \boldsymbol{X}) = \omega\left(\rho\left(\boldsymbol{A}, \boldsymbol{X}\right)\right) \tag{2.22}$$

combining k iterations of message-passing with the classifier presented in Equation (2.12). Here, ρ is a permutation equivariant function and ω is a permutation invariant function.

While many choices for the function ρ and individual message-passing functions $g_1, \ldots, g_k$ have been proposed, finding the optimal choices and identifying their limitations or beneficial properties is a key aspect of many research efforts. We will discuss various choices and their properties in the following chapters.

Further Machine Learning Tasks on Graphs

Apart from the graph classification task, which aims to predict a single value for each graph, many other tasks are of interest. Predicting values for individual nodes is of interest for many applications. For example, when a graph corresponds to a road network, we might be interested in predicting traffic at multiple locations (Roth and Liebig 2022a). These node-level tasks include node classification and node regression, where a discrete class or a continuous value is predicted for every node, respectively. The graph convolution and MPNNs can be easily adapted to such tasks.

By applying a permutation equivariant function $\rho(\boldsymbol{A}, \boldsymbol{X})$, the updated node representations $\boldsymbol{X}' = \rho(\boldsymbol{A}, \boldsymbol{X})$ can contain information about the structural properties of each node and their combination with node attributes. The vector of representations $\boldsymbol{X}'_{i,:}$ of each node i is then used to perform node-level prediction tasks. That means, based on a function $h\colon \mathbb{R}^d \to \mathcal{Y}$, we can assign a class or continuous prediction $\hat{y}_{(i)} = h\left(\boldsymbol{X}'_{i,:}\right)$ to every node v_i.

Similarly to these node-level prediction tasks, edge-level prediction tasks can be performed. For each node combination (v_i, v_j) of interest, a classifier $e\colon \mathbb{R}^{d'} \times \mathbb{R}^{d'} \to \mathcal{Y}$ can be applied to the representations of the two nodes, i.e., $\hat{\boldsymbol{y}}_{(i,j)} = e\left(\boldsymbol{X}'_{i,:}, \boldsymbol{X}'_{j,:}\right)$. Many other tasks are similarly defined based on transformed node representations $\boldsymbol{X}' = \rho\left(\boldsymbol{A}, \boldsymbol{X}\right)$. This concludes the foundational background required for the following sections.

3 Extending The Understanding of Graph Convolutions

Many message-passing neural networks (MPNNs) exhibit a degradation of performance on several tasks as the number of message-passing iterations increases. Several related phenomena - such as over-smoothing, over-correlation, over-squashing, and vanishing gradients - have been proposed, but their interrelations and underlying theoretical foundations remain poorly understood. In this chapter, we build upon prior work on over-smoothing and present an in-depth theoretical analysis of widely used MPNNs. We identify two key properties of a commonly used class of MPNNs that fundamentally limit their performance. The first, shared component amplification, refers to the same component of the representations being amplified in all feature channels during each iteration. The second, component dominance, describes how, with an increased number of iterations, a single component of the node representations gets dominantly amplified while all other components are progressively suppressed. Our analysis provides a deeper theoretical and intuitive understanding of several known phenomena, enabling the development of methods that avoid these properties. We validate our analysis through a series of experiments that empirically confirm our theoretical findings and expand upon them.

3.1 Introduction

As localized approximations of the spectral graph convolution, message-passing neural networks (MPNNs) have achieved promising results in various applications, such as traffic forecasting (Roth and Liebig 2022a) or molecular prediction (Stärk et al. 2022). However, it has also been observed that the performance of MPNNs degrades for various tasks when increasing the number of message-passing iterations (Q. Li et al. 2018). For many tasks, the best performance is achieved after only two such iterations (Kipf et al. 2017). For other tasks, a single iteration is already detrimental to the performance (Rusch, B. Chamberlain, et al. 2022). Initially, this performance degradation was observed for the graph convolutional network (GCN) (Kipf et al. 2017), with various other methods exhibiting similar degradation (Q. Li et al. 2018; NT et al. 2019).

Studying the causes behind this performance degradation is crucial for multiple reasons. Many tasks require the combination of representations that are not directly connected or even far apart within a graph (Gilmer et al. 2017; Alon et al. 2021). Multiple iterations also allow for more feature transformations and non-linearities, potentially allowing for more complex features. A better understanding also ideally provides insights into the severity of this phenomenon for the performance of shallow MPNNs with few iterations.

Many previous works studied this performance degradation empirically or theoretically, such as over-smoothing (Q. Li et al. 2018; NT et al. 2019; Oono et al. 2020; Cai et al. 2020; Giovanni et al. 2023; Rusch, Bronstein, et al. 2023), over-squashing (Alon et al. 2021; Topping et al. 2022), and over-correlation (Jin et al. 2022). However, the theoretical details of these phenomena remain poorly understood. As a result, current research on performance degradation is scattered and many different approaches to mitigate aspects of these phenomena have been proposed. Several works attribute the performance degradation to a phenomenon referred to as over-smoothing, which refers to the representations of different nodes becoming increasingly similar with an increased number of layers (Q. Li et al. 2018; Oono et al. 2020). Understanding the mathematical properties of MPNNs leading to over-smoothing has been an active area of research (Q. Li et al. 2018; NT et al. 2019; Oono et al. 2020; Cai et al. 2020). However, previous studies have not managed to provide comprehensive theoretical insights, even for linearized cases. Multiple different definitions and intuitions for over-smoothing currently coexist, with the role of learnable parameters remaining unclear. Therefore, current methods to mitigate over-smoothing are built on an incomplete theoretical understanding and tackle different aspects of the phenomenon. It is unclear which of the many aspects related to the performance degradation are most promising for focusing future research on.

In this chapter, we improve the theoretical understanding of MPNNs. In Section 3.2, we provide an extensive overview of current insights into over-smoothing and over-correlation. We include both empirical and theoretical studies. In Section 3.2.2, we conduct an empirical analysis of convolutional filters in the graph Fourier domain. As MPNNs were derived as localized approximations of the spectral graph convolution defined using the graph Fourier transform, we compare their expressible convolutional filters. To clarify conflicting definitions of over-smoothing in previous work, we find that a vanishing norm of representations overshadows potential insights of previous studies in Section 3.3.3.

In Section 3.5, we present our main theoretical study. We identify two key properties of a general class of MPNNs. We refer to the first property as shared component amplification (SCA) that describes a property of a single message-passing iteration. We find that the same component gets amplified for any feature transformation and across all feature channels. As the second property, we identify that a single component is increasingly dominantly amplified, with all other components being relatively filtered out. We term this property component dominance (CD). These properties lead to a phenomenon that we call rank collapse, where representations become closer to a lower-dimensional subspace. We further show that the observed phenomenon of over-smoothing occurs as a special case for particular aggregation functions. We also introduce a property that allows message-passing methods to prevent the underlying phenomena. This property requires message-passing methods to use multiple distinct aggregation functions and feature transformations.

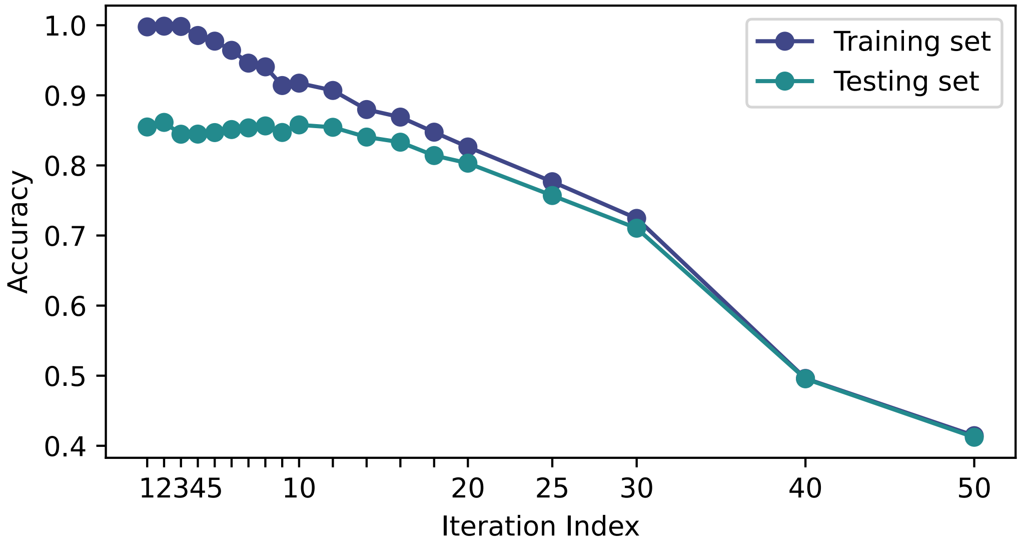


Figure 3.1: Training and testing accuracy achieved on the Cora (Sen et al. 2008) dataset for training GCN models with different numbers of iterations. This experiment is based on Kipf et al. (2017).

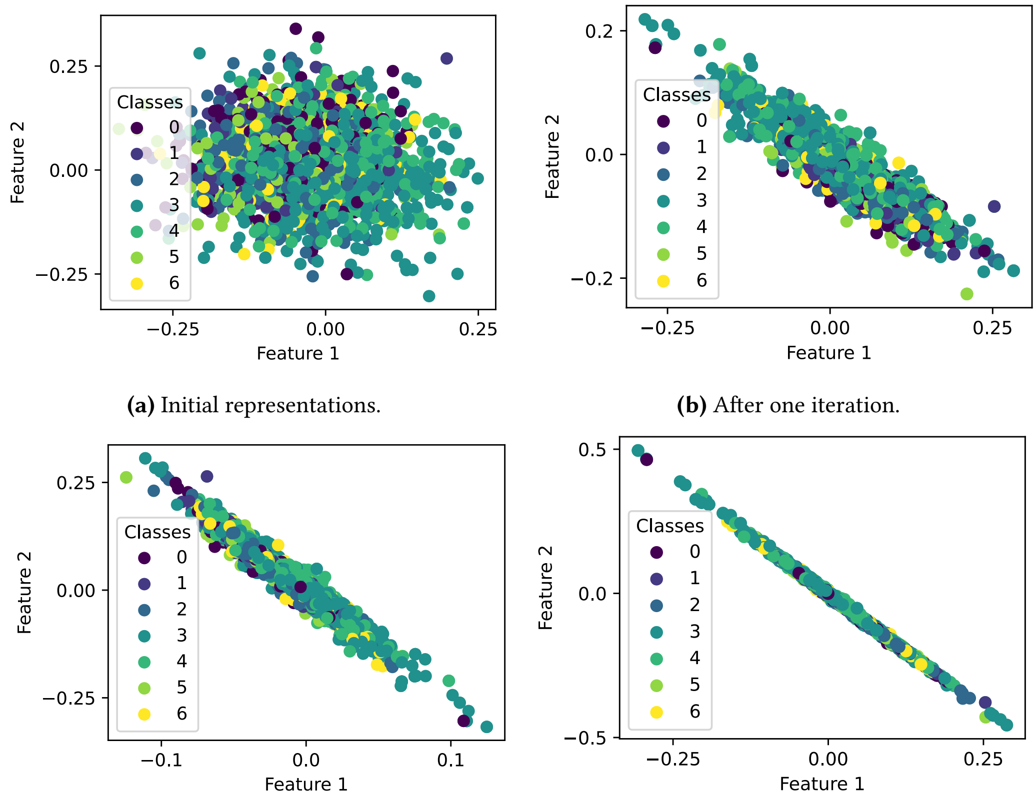
In Section 3.6, we summarize our extensive findings, provide definitions for the varying phenomena, and show how over-smoothing and rank collapse can be prevented by methods that directly change the underlying properties. We empirically support our theoretical findings with various experiments in Section 3.7.

3.2 Related Work

While a consensus in previous work on over-smoothing is that node representations become more similar with an increasing number of iterations (Q. Li et al. 2018; Oono et al. 2020; Rusch, Bronstein, et al. 2023), the details are less clear. Here, we detail the available insights into over-smoothing from previous works. We review available studies and provide proofs of key theoretical statements that our work builds upon.

Most theoretical studies consider an attributed graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ in matrix form, where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix representing a graph with n nodes, and $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ is the matrix containing d attributes for each node. Unless otherwise indicated, we assume $\mathcal{G}$ is an undirected graph that is connected and non-bipartite. Previous works also mainly consider the graph convolutional network (GCN) (Kipf et al. 2017) as an approximation to the spectral graph convolution. A message-passing iteration of the GCN in matrix notation is defined as

$$\boldsymbol{X}^{(k)} = \sigma\left(\boldsymbol{A}_{\text{sym}} \boldsymbol{X}^{(k-1)} \boldsymbol{W}^{(k)}\right) , \tag{3.1}$$



(a) Initial representations. **(b)** After one iteration.

(c) After two iterations. **(d)** After three iterations.

Figure 3.2: Two-dimensional node features $\boldsymbol{X}^{(k)} \in \mathbb{R}^{n \times d}$ after applying up to three GCN iterations $\boldsymbol{X}^{(k)} = \boldsymbol{A}_{\text{sym}} \boldsymbol{X}^{(k-1)} \boldsymbol{W}^{(k)}$ where $\boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{n \times n}$ is given by the Cora dataset (Sen et al. 2008). $\boldsymbol{X}^{(0)}$ is given by a linear transformation of the provided node features for Cora. All parameters are randomly initialized. Each dot represents the state of a single node. This experiment is based on Q. Li et al. (2018).

where the initial node representations $\boldsymbol{X}^{(0)}$ are typically node-wise obtained by applying a node-wise transformation to $\boldsymbol{X}$, $\boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{n \times n}$ is the symmetrically normalized adjacency matrix. The matrix $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d \times d}$ contains the learnable parameters and serves as a feature transformation. σ is a non-linear activation function, typically assumed to be the identity function for theoretical analysis.

3.2.1 Over-Smoothing

Here, we provide an extensive review of available insights into over-smoothing. We also present proofs of the theoretical findings underlying our subsequent analysis. When the GCN was initially proposed (Kipf et al. 2017), the authors conducted an ablation study that showed the optimal performance was achieved after two iterations of the GCN. Further iterations were shown to decrease performance for both the training and testing sets. To

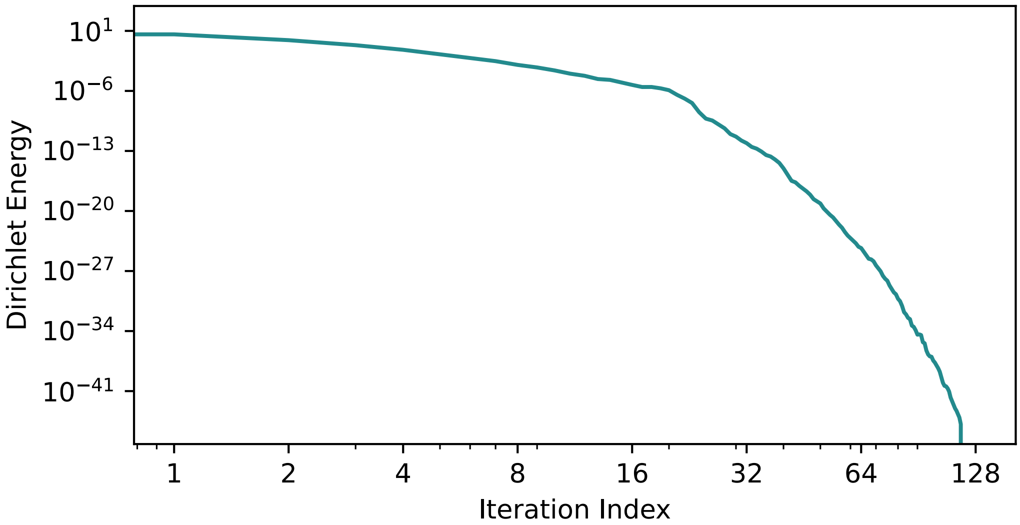


Figure 3.3: Dirichlet energy $E\left(\boldsymbol{X}^{(k)}\right) = \operatorname{tr}\left(\left(\boldsymbol{X}^{(k)}\right)^{\top} \boldsymbol{L}_{\mathrm{sym}} \boldsymbol{X}^{(k)}\right)$ for $\boldsymbol{X}^{(k)} = \phi\left(\boldsymbol{A}_{\mathrm{sym}} \boldsymbol{X}^{(k-1)} \boldsymbol{W}^{(k)}\right)$ where ϕ is the ReLU activation function, each $\boldsymbol{W}^{(k)}$ is randomly initialized and k indicates the iteration number. The Cora dataset was used for initial features and as graph structure (Sen et al. 2008). Average values over 20 random initializations are shown. This experiment is based on Oono et al. (2020) and Cai et al. (2020).

provide an initial visual intuition for this performance degradation of the GCN with an increasing number of iterations, we reproduce their findings and visualize the achieved training and testing accuracy in Figure 3.1. It confirms that the obtained accuracy for both the training and testing sets degrades with an increased number of iterations. As both sets exhibit similar performance degradation, Kipf et al. (2017) argue that overfitting is not the primary cause behind this phenomenon.

The Effect of the Aggregation Q. Li et al. (2018) initially introduced the term over-smoothing by studying limitations of the GCN. First, they visualized two-dimensional node representations after applying a varying number of randomly initialized GCN iterations. We reproduce their experiment and provide the visualization in Figure 3.2. They empirically observed that node representations become increasingly aligned with more iterations.

They theoretically studied the effect of the aggregation step $\boldsymbol{A}_{\mathrm{sym}}$ of the GCN without the feature transformation and non-linear activation function. They relate the aggregation to Laplacian smoothing of node representations and state the following key theorem:

Theorem 3.1 (Theorem 1 from Q. Li et al. (2018)). Let $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ be the adjacency matrix of an undirected and non-bipartite graph. Then, for any $\boldsymbol{x} \in \mathbb{R}^n$,

$$\lim_{k\to\infty} (\boldsymbol{A}_{\text{sym}})^k \boldsymbol{x} = c \cdot \boldsymbol{D}^{-1/2}\mathbf{1} \tag{3.2}$$

$$\lim_{k\to\infty} (\boldsymbol{A}_{\text{rw}})^k \boldsymbol{x} = d \cdot \mathbf{1} \tag{3.3}$$

where $\mathbf{1} \in \mathbb{R}^n$ and $c, d \in \mathbb{R}$.

Proof. We will start with the proof for $\boldsymbol{A}_{\text{sym}}$ and show the necessary changes for $\boldsymbol{A}_{\text{rw}}$ afterwards. Consider the decomposition $\boldsymbol{x} = b_1\boldsymbol{U}_{:,1} + \cdots + b_n\boldsymbol{U}_{:,n}$ of $\boldsymbol{x}$ as a linear combination of the eigenvectors $\boldsymbol{U}$ of $\boldsymbol{A}_{\text{sym}}$ with coefficients $\boldsymbol{b} = \boldsymbol{U}^\top\boldsymbol{x} \in \mathbb{R}^n$. Applying $\boldsymbol{A}_{\text{sym}}$ to the decomposed form results in $\boldsymbol{A}_{\text{sym}}\boldsymbol{x} = \boldsymbol{A}_{\text{sym}}b_1\boldsymbol{U}_{:,1} + \cdots + \boldsymbol{A}_{\text{sym}}b_n\boldsymbol{U}_{:,n}$. Each $\boldsymbol{U}_{:,i}$ is an eigenvector of $\boldsymbol{A}_{\text{sym}}$ with corresponding eigenvalue λ_i by definition. This simplifies the application of $\boldsymbol{A}_{\text{sym}}$ to

$$\boldsymbol{A}_{\text{sym}}\boldsymbol{x} = \lambda_1 b_1\boldsymbol{U}_{:,1} + \cdots + \lambda_n b_n\boldsymbol{U}_{:,n} \,. \tag{3.4}$$

Following properties known from spectral graph theory (Section 2.2.2), the eigenvalues satisfy the properties $\lambda_1 = 1$ and $|\lambda_i| < 1$ for all $i > 1$ and any connected non-bipartite undirected graph. In the limit case, this simplifies to

$$\lim_{k\to\infty} \lambda_i^k = \begin{cases} 1, & \text{if } i = 1 \\ 0, & \text{otherwise} \,. \end{cases} \tag{3.5}$$

Combining these properties, we have

$$\lim_{k\to\infty} \boldsymbol{A}_{\text{sym}}^k\boldsymbol{x} = b_1\boldsymbol{U}_{:,1} \,. \tag{3.6}$$

As known in spectral graph theory (Section 2.2.2), the corresponding eigenvector $\boldsymbol{U}_{:,1}$ of $\boldsymbol{A}_{\text{sym}}$ is a degree-proportional vector $\boldsymbol{D}^{1/2}\mathbf{1}$.

The proof for $\boldsymbol{A}_{\text{rw}}$ is equivalent, as only the eigenvectors and eigenvalues change. For $\boldsymbol{A}_{\text{rw}}$, we also have $\lambda_1 = 1$ and $|\lambda_i| < 1$ for $i > 1$. It is also known that any $\boldsymbol{A}_{\text{rw}}$ under the above assumptions has eigenvector $\boldsymbol{U}_{:,1} = \mathbf{1}$ (Section 2.2.2). □

Recalling from Section 2.2.2, we refer to $\boldsymbol{b} = \boldsymbol{U}^\top\boldsymbol{x}$ as the (spectral) components of $\boldsymbol{x}$. We denote the component of $\boldsymbol{x}$ corresponding to eigenvector $\boldsymbol{U}_{:,i}$ by b_i. Q. Li et al. (2018) find components corresponding to eigenvectors of $\boldsymbol{A}_{\text{sym}}$ to be amplified based on their respective eigenvalue. They further argue that smoothing is a key reason for the success of the GCN as the smoothing property can ease classification tasks. At the same time, they argue that excessive smoothing combines representations that should not be similar, making optimization more challenging. A similar study was later conducted by Keriven (Keriven 2022), showing that some smoothing is beneficial for some downstream tasks, while excessive iterations are detrimental. NT et al. (2019) show that the aggregations $\boldsymbol{A}_{\text{sym}}$ and $\boldsymbol{A}_{\text{rw}}$ act like low-pass filters that suppress high-frequency components from the node representations, while low-frequency components are less affected. For other aggregation functions $\tilde{\boldsymbol{A}}$, it has been shown that high-frequency components can be amplified (Bo et al. 2021; Luan et al. 2022).

The Effect of the Feature Transformation While the effect of aggregating features with $\boldsymbol{A}_{\text{sym}}$ or $\boldsymbol{A}_{\text{rw}}$ is understood, the impact of feature transformations was not considered by Q. Li et al. (2018). Oono et al. (2020) consider the distance of node representations to a smooth subspace spanned by the eigenvector $\boldsymbol{U}_{:,1}$ corresponding to the largest eigenvalue of $\boldsymbol{A}_{\text{sym}}$ in magnitude. When applying a feature transformation $\boldsymbol{W}^{(k)}$ to node representations, they prove that the distance of the resulting node representations to this subspace can be upper bounded by the largest singular value σ_1 of $\boldsymbol{W}^{(k)}$. When σ_1 is sufficiently small, representations converge exponentially fast in the number of iterations to the smooth subspace.

Cai and Wang (Cai et al. 2020) propose to utilize the graph Dirichlet energy as a metric to quantify over-smoothing. It is defined as

$$\begin{aligned} E(\boldsymbol{X}) &= \operatorname{tr}\left(\boldsymbol{X}^\top \boldsymbol{L}_{\text{sym}} \boldsymbol{X}\right) \\ &= \frac{1}{2} \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{N}_i} \left\| \frac{\boldsymbol{X}_{i,:}}{\sqrt{d_i}} - \frac{\boldsymbol{X}_{j,:}}{\sqrt{d_j}} \right\|_2^2 \end{aligned} \tag{3.7}$$

as detailed in Section 2.2.2. Equation (3.7) is the spatial form of the Dirichlet energy. Due to its origin in spectral graph theory, the Dirichlet energy has an intuitive spectral interpretation. The graph Laplacian can be decomposed into $\boldsymbol{L}_{\text{sym}} = \boldsymbol{U}\left(\mathbf{I}_n - \boldsymbol{\Lambda}\right)\boldsymbol{U}^\top$, where $\boldsymbol{U}$ is the orthonormal matrix of eigenvectors and $\boldsymbol{\Lambda}$ is the diagonal matrix of eigenvalues of $\boldsymbol{A}_{\text{sym}}$. The null space of $\boldsymbol{L}_{\text{sym}}$ is spanned by the eigenvector $\boldsymbol{U}_{:,1} = \boldsymbol{D}^{1/2}\mathbf{1}$ with corresponding eigenvalue 0, i.e., $\boldsymbol{L}_{\text{sym}}\boldsymbol{D}^{1/2}\mathbf{1} = \mathbf{0}$. In the graph Fourier domain, this vector corresponds to the lowest-frequency (i.e., the smoothest) component. The node representations can be expressed in the graph Fourier space as $\boldsymbol{B} = \boldsymbol{U}^\top \boldsymbol{X}$. For every feature channel $c \in [d]$, $B_{i,c}$ indicates the strength of its component corresponding to eigenvector $\boldsymbol{U}_{:,i}$. As $\boldsymbol{U}$ is orthonormal, we have $\boldsymbol{U}\boldsymbol{U}^\top = \boldsymbol{I}_n$. This leads to the equivalent spectral form of the Dirichlet energy

$$E(\boldsymbol{X}) = \sum_{i=1}^{n} \sum_{c=1}^{d} \left(1 - \lambda_i\right) B_{i,c}^2 \,. \tag{3.8}$$

As $\lambda_1 = 1$, we have $1 - \lambda_1 = 0$, and thus all components apart from the components of $\boldsymbol{X}$ corresponding to eigenvector $\boldsymbol{U}_{:,1}$ are taken into account. When $\boldsymbol{X}$ consists of non-zero components only corresponding to $\boldsymbol{U}_{:,1}$, the Dirichlet energy is zero. Cai and Wang (Cai et al. 2020) study the effect of GCN iterations on the Dirichlet energy and state the following key proposition:

Proposition 3.1 (Proposition 3.4 from Cai et al. (2020)). Let $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ be any matrix, ϕ be the ReLU or LeakyReLU activation function, and λ_2 be the second largest eigenvalue in magnitude of $\boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{n \times n}$. Further, let $\boldsymbol{W} \in \mathbb{R}^{d \times d'}$ be any matrix for some d' with σ_1 being its largest singular value in magnitude. Then,

$$E\left(\phi\left(\boldsymbol{A}_{\text{sym}}\boldsymbol{X}\boldsymbol{W}\right)\right) \leq \sigma_1^2 \cdot \lambda_2 \cdot E(\boldsymbol{X})\,. \tag{3.9}$$

Proof. Equivalently to (Cai et al. 2020, Lemma 3.1), we first prove the inequality

$$E\left(\boldsymbol{A}_{\text{sym}}\boldsymbol{X}\right) \leq \lambda_2 \cdot E(\boldsymbol{X})\,. \tag{3.10}$$

Consider the decomposition $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{B}$, where $\boldsymbol{B} = \boldsymbol{U}^\top \boldsymbol{X}$ is the graph Fourier transformed representation. We use the fact that $\boldsymbol{A}_{\text{sym}}$ and $\boldsymbol{L}$ have the same orthonormal eigenvectors $\boldsymbol{U}$. Writing the Dirichlet energy in the spectral form results in

$$E\left(\boldsymbol{A}_{\text{sym}}\boldsymbol{X}\right) = \operatorname{tr}\left(\boldsymbol{B}^\top\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top\boldsymbol{U}\left(\boldsymbol{I}_n - \boldsymbol{\Lambda}\right)\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{B}\right)\,. \tag{3.11}$$

As $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}_n$, this simplifies to

$$E\left(\boldsymbol{A}_{\text{sym}}\boldsymbol{X}\right) = \operatorname{tr}\left(\boldsymbol{B}^\top\boldsymbol{\Lambda}\left(\boldsymbol{I}_n - \boldsymbol{\Lambda}\right)\boldsymbol{\Lambda}\boldsymbol{B}\right)\,. \tag{3.12}$$

As $\boldsymbol{\Lambda}$ and $\boldsymbol{I}_n$ are diagonal matrices, this further simplifies as a sum of scalars:

$$E\left(\boldsymbol{A}_{\text{sym}}\boldsymbol{X}\right) = \sum_{i=1}^{n}\sum_{c=1}^{d}\left(1-\lambda_i\right)\lambda_i^2 B_{i,c}^2 \tag{3.13}$$

As $1 - \lambda_1 = 0$, this can then be upper bounded by

$$E\left(\boldsymbol{A}_{\text{sym}}\boldsymbol{X}\right) \leq \lambda_2^2 \cdot E(\boldsymbol{X}) \tag{3.14}$$

Next, we show that $E(\boldsymbol{X}\boldsymbol{W}) \leq \sigma_1^2 E(\boldsymbol{X})$ (Cai et al. 2020, Lemma 3.2) using the spatial form of the Dirichlet energy:

$$E(\boldsymbol{X}\boldsymbol{W}) = \frac{1}{2}\sum_{v_i\in\mathcal{V}}\sum_{v_j\in\mathcal{N}_i}\left\|\frac{\boldsymbol{X}_{i,:}\boldsymbol{W}}{\sqrt{d_i}} - \frac{\boldsymbol{X}_{j,:}\boldsymbol{W}}{\sqrt{d_j}}\right\|_2^2\,. \tag{3.15}$$

By sub-multiplicativity of the norm, this is bounded by

$$E(\boldsymbol{X}\boldsymbol{W}) \leq \frac{1}{2}\sum_{v_i\in\mathcal{V}}\sum_{v_j\in\mathcal{N}_i}\left\|\frac{\boldsymbol{X}_{i,:}}{\sqrt{d_i}} - \frac{\boldsymbol{X}_{j,:}}{\sqrt{d_j}}\right\|_2^2 \|\boldsymbol{W}\|_2^2\,. \tag{3.16}$$

By definition of the Dirichlet energy, we thus have

$$E(\boldsymbol{X}\boldsymbol{W}) \leq E(\boldsymbol{X})\|\boldsymbol{W}\|_2^2\,. \tag{3.17}$$

As the 2-norm $\|\boldsymbol{W}\|_2 = \sigma_1$ is equal to the largest singular value of $\boldsymbol{W}$, it follows that

$$E(\boldsymbol{X}\boldsymbol{W}) \leq \sigma_1^2 \cdot E(\boldsymbol{X})\,. \tag{3.18}$$

To show that $E\left(\phi\left(\boldsymbol{X}\right)\right) \leq E(\boldsymbol{X})$, the property that ReLU and LeakyReLU are 1-Lipschitz is used, i.e., $\|\phi(\boldsymbol{P}) - \phi(\boldsymbol{Q})\| \leq \|\boldsymbol{P} - \boldsymbol{Q}\|$ for any matrices $\boldsymbol{P}, \boldsymbol{Q}$. These activation functions also satisfy that for any non-negative scalar r, it holds that $\phi(r \cdot \boldsymbol{P}) = r \cdot \phi(\boldsymbol{P})$.

Applying these two properties to the Dirichlet energy proves this part (Cai et al. 2020, Lemma 3.3):

$$
\begin{aligned}
E\left(\phi(\boldsymbol{X})\right) &= \frac{1}{2}\sum_{v_i\in\mathcal{V}}\sum_{v_j\in\mathcal{N}_i}\left\|\frac{\phi\left(\boldsymbol{X}_{i,:}\right)}{\sqrt{d_i}}-\frac{\phi\left(\boldsymbol{X}_{j,:}\right)}{\sqrt{d_j}}\right\|_2^2 \\
&= \frac{1}{2}\sum_{v_i\in\mathcal{V}}\sum_{v_j\in\mathcal{N}_i}\left\|\phi\left(\frac{\boldsymbol{X}_{i,:}}{\sqrt{d_i}}\right)-\phi\left(\frac{\boldsymbol{X}_{j,:}}{\sqrt{d_j}}\right)\right\|_2^2 \\
&\leq \frac{1}{2}\sum_{v_i\in\mathcal{V}}\sum_{v_j\in\mathcal{N}_i}\left\|\frac{\boldsymbol{X}_{i,:}}{\sqrt{d_i}}-\frac{\boldsymbol{X}_{j,:}}{\sqrt{d_j}}\right\|_2^2 = E(\boldsymbol{X})
\end{aligned}
\tag{3.19}
$$

Combining these three properties concludes this proof. □

Based on Proposition 3.1, the Dirichlet energy is decreasing in each iteration when $\sigma_1 < \frac{1}{\lambda_2}$. We visualize the Dirichlet energy converging to zero with an increased number of iterations in Figure 3.3.

As the Dirichlet energy can be upper-bounded by the singular values of the feature transformation $\boldsymbol{W}$, Zhou et al. (2021) showed that the Dirichlet energy can similarly be lower-bounded based on the smallest singular value. They refer to the phenomenon when the Dirichlet energy becomes excessively large as an over-separation of node representations.

Normalization and Over-Separation Contrarily, Giovanni et al. (2023) found that a large Dirichlet energy does not necessarily imply an over-separation of node representations. They consider the case

$$
\boldsymbol{X}^{(k)} = \boldsymbol{A}_{\text{sym}}\boldsymbol{X}^{(k-1)}\boldsymbol{W}
\tag{3.20}
$$

without a non-linear activation function where $\boldsymbol{W} \in \mathbb{R}^{d\times d}$ is a symmetric matrix that is shared across iterations. As shown by (Zhou et al. 2021), the Dirichlet energy of $\boldsymbol{X}^{(k)}$ may grow unboundedly $\lim_{k\to\infty} E(\boldsymbol{X}^{(k)}) = \infty$. Giovanni et al. (2023) find that when considering the normalized node representations, the Dirichlet energy still converges to zero, i.e., $\lim_{k\to\infty} E\left(\frac{\boldsymbol{X}^{(k)}}{\|\boldsymbol{X}^{(k)}\|_F}\right) = 0$ for Equation (3.20). Because the smooth component dominates all other components, they refer to this phenomenon as low-frequency dominated dynamics. They prove this is the case for Equation (3.20) and any symmetric feature transformation $\boldsymbol{W} \in \mathbb{R}^{d\times d}$, independently of its singular values:

Proposition 3.2 (Theorem 5.3 from Giovanni et al. (2023)). Let $\boldsymbol{A} \in \{0,1\}^{n\times n}$ be the adjacency matrix of an undirected and non-bipartite graph and $\boldsymbol{W} \in \mathbb{R}^{d\times d}$ be a symmetric matrix. Then, for a.e. matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$,

$$\lim_{k\to\infty} E\left(\frac{\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k}{\left\|\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k\right\|_F}\right) = 0\,. \tag{3.21}$$

Proof. We present an alternative proof to the one used by Giovanni et al. (2023), which aligns better with our subsequent theoretical analysis.

We start by using the definition of the Dirichlet energy and the Frobenius norm:

$$E\left(\frac{\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k}{\left\|\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k\right\|_F}\right) = \frac{\operatorname{tr}\left(\boldsymbol{W}^k\boldsymbol{X}^\top \boldsymbol{A}_{\text{sym}}^k \boldsymbol{L}_{\text{sym}} \boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k\right)}{\operatorname{tr}\left(\boldsymbol{W}^k\boldsymbol{X}^\top \boldsymbol{A}_{\text{sym}}^k \boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k\right)} \tag{3.22}$$

Using the property $\operatorname{tr}(\boldsymbol{P}\top\boldsymbol{Q}) = \operatorname{vec}(\boldsymbol{P})^\top \operatorname{vec}(\boldsymbol{Q})$ for any suitable matrices $\boldsymbol{P},\boldsymbol{Q}$ we have

$$E\left(\frac{\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k}{\left\|\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k\right\|_F}\right) = \frac{\operatorname{vec}\left(\boldsymbol{L}_{\text{sym}}\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k\right)^\top \operatorname{vec}\left(\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}^\top\boldsymbol{W}^k\right)}{\operatorname{vec}\left(\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k\right)^\top \operatorname{vec}\left(\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}^\top\boldsymbol{W}^k\right)} \tag{3.23}$$

Further using the property $\operatorname{vec}(\boldsymbol{P}\boldsymbol{Q}\boldsymbol{R}) = (\boldsymbol{R}^\top \otimes \boldsymbol{P})\operatorname{vec}(\boldsymbol{Q})$ for any suitable matrices $\boldsymbol{P},\boldsymbol{Q},\boldsymbol{R}$,

$$E\left(\frac{\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k}{\left\|\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k\right\|_F}\right) = \frac{\operatorname{vec}(\boldsymbol{X})^\top \left(\boldsymbol{W}^k \otimes \boldsymbol{L}_{\text{sym}}\boldsymbol{A}_{\text{sym}}^k\right)^\top \left(\boldsymbol{W}^k \otimes \boldsymbol{A}_{\text{sym}}^k\right)\operatorname{vec}(\boldsymbol{X})}{\operatorname{vec}(\boldsymbol{X})^\top \left(\boldsymbol{W}^k \otimes \boldsymbol{A}_{\text{sym}}^k\right)^\top \left(\boldsymbol{W}^k \otimes \boldsymbol{A}_{\text{sym}}^k\right)\operatorname{vec}(\boldsymbol{X})} \tag{3.24}$$

Now let $\boldsymbol{A}_{\text{sym}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$ and $\boldsymbol{W} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ be the eigendecompositions of $\boldsymbol{A}_{\text{sym}}$ and $\boldsymbol{W}$. We then restate $\operatorname{vec}(\boldsymbol{X}) = (\boldsymbol{V}\otimes\boldsymbol{U})\operatorname{vec}(\boldsymbol{B})$ using the eigenbasis $\boldsymbol{V}\otimes\boldsymbol{U}$ where $\operatorname{vec}(\boldsymbol{B}) = (\boldsymbol{V}\otimes\boldsymbol{U})^\top \operatorname{vec}(\boldsymbol{X})$. This simplifies to

$$E\left(\frac{\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k}{\left\|\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k\right\|_F}\right) = \frac{\operatorname{vec}(\boldsymbol{B})^\top \left(\boldsymbol{\Sigma}^{2k} \otimes \boldsymbol{\Lambda}^k \left(\boldsymbol{I}_n - \boldsymbol{\Lambda}\right)\boldsymbol{\Lambda}^{2k}\right)\operatorname{vec}(\boldsymbol{B})}{\operatorname{vec}(\boldsymbol{B})^\top \left(\boldsymbol{\Sigma} \otimes \boldsymbol{\Lambda}^{2k}\right)\operatorname{vec}(\boldsymbol{B})} \tag{3.25}$$

Due to diagonality of $\boldsymbol{\Sigma}^{2k} \otimes \boldsymbol{\Lambda}^k \left(\boldsymbol{I}_n - \boldsymbol{\Lambda}\right)\boldsymbol{\Lambda}^k$ this simplifies to

$$E\left(\frac{\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k}{\left\|\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k\right\|_F}\right) = \frac{\sum_{i=1}^n \sum_{r=1}^d B_{i,r}^2 \lambda_i^{2k} \sigma_r^{2k} \left(1-\lambda_i\right)}{\sum_{i=1}^n \sum_{r=1}^d B_{i,r}^2 \lambda_i^{2k} \sigma_r^{2k}}\,. \tag{3.26}$$

We normalize the numerator and the denominator by dividing by the largest absolute eigenvalue σ_1^{2k} of the parameter matrix. This leads to the identity

$$E\left(\frac{\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k}{\left\|\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k\right\|_F}\right) = \frac{\sum_{i=1}^n \sum_{r=1}^d B_{i,r}^2 \lambda_i^{2k} \left(\frac{\sigma_r^{2k}}{\sigma_1^{2k}}\right)(1-\lambda_i)}{\sum_{i=1}^n \sum_{r=1}^d B_{i,r}^2 \lambda_i^{2k} \left(\frac{\sigma_r^{2k}}{\sigma_1^{2k}}\right)}. \tag{3.27}$$

As $\left|\lambda_i \left(\frac{\sigma_r}{\sigma_1}\right)\right| < 1$ for $i > 1$, the, the numerator decays to zero as $k \to \infty$ and we have

$$\lim_{k\to\infty} E\left(\frac{\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k}{\left\|\boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}\boldsymbol{W}^k\right\|_F}\right) = 0 \tag{3.28}$$

assuming $B_{i,1} \neq 0$, which is satisfied for a.e. choice for $\boldsymbol{X}$.

□

Based on this finding, Giovanni et al. (2023) also show cases in which a different frequency component can dominate the node representation. Maskey et al. (2023) build on this idea and show that the Dirichlet energy of the normalized representations also converges to zero for directed graphs when using the symmetrically normalized adjacency matrix. Both of these works assume $\boldsymbol{W}$ to be a symmetric matrix that is shared across iterations.

Convergence to a Constant State Rusch, B. Chamberlain, et al. (2022) empirically observed that $\boldsymbol{X}^{(k)}$ converges to a constant state for the GCN and various other established methods. We reproduced their experiments and visualize this phenomenon for several methods in Figure 3.4. In follow-up works, over-smoothing was defined accordingly as a convergence to a constant state (Rusch, B. P. Chamberlain, et al. 2023; Rusch, Bronstein, et al. 2023). Their definition utilizes a node similarity measure that is zero when all node representations are equal. Over-smoothing is then defined as an exponential convergence to this state over the number of iterations. Many of the following studies and methods were built on this definition (Wu et al. 2023; Kothapalli et al. 2023).

This observation does not align with Theorem 3.1 and Proposition 3.2, as these theoretical insights show that $\boldsymbol{X}^{(k)}$ converges to a degree-proportional state for some of the same methods. It is currently unclear how these similar yet slightly different observations relate to one another. It also remains unclear whether the over-separation of node representations is a relevant phenomenon. Similarly, the complete theoretical implications of applying arbitrary feature transformations remain an open question. As the theoretical analyses mainly considered $\boldsymbol{A}_{\text{sym}}$ as the aggregation matrix, the effect of further aggregation functions $\tilde{\boldsymbol{A}}$ or specific choices for edge weights is also not fully understood. Due to this incomplete understanding of over-smoothing and related phenomena, potential solutions to over-smoothing may address different aspects behind the umbrella term over-smoothing and may not address the underlying problem.

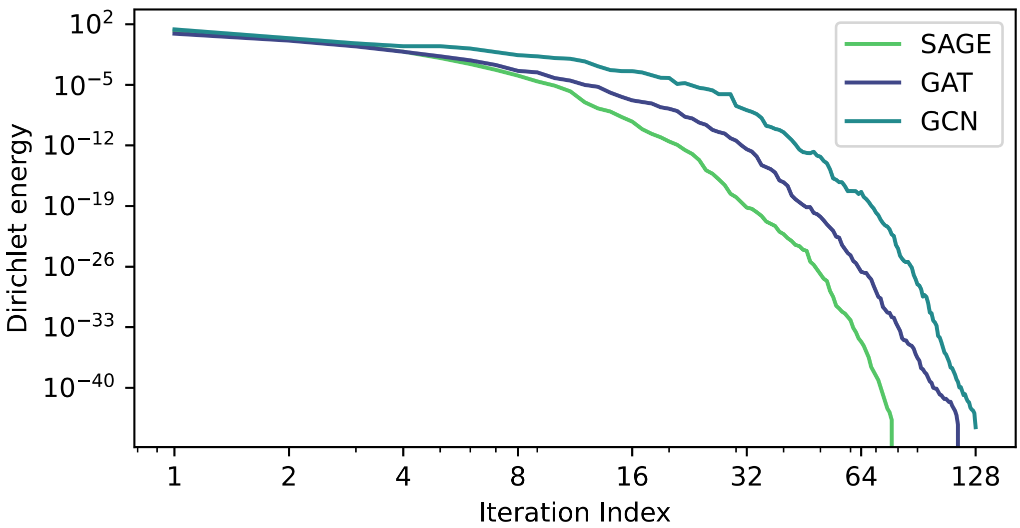


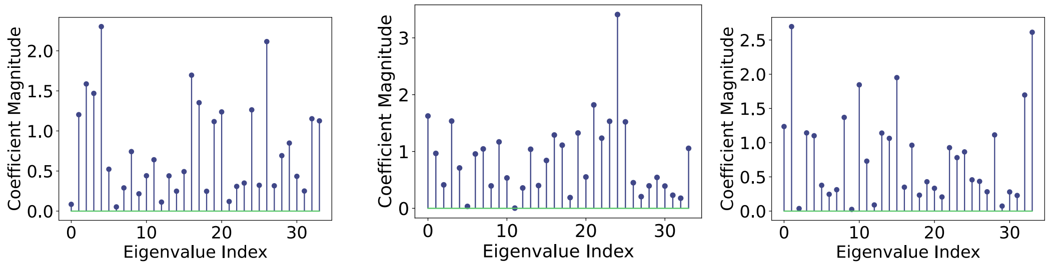
Figure 3.4: Dirichlet energy $E\left(\boldsymbol{X}^{(k)}\right) = \operatorname{tr}\left(\left(\boldsymbol{X}^{(k)}\right)^\top \boldsymbol{L}\boldsymbol{X}^{(k)}\right)$ using the unnormalized graph Laplacian $\boldsymbol{L}$ for the node representations obtained by three message-passing methods on the Cora dataset (Sen et al. 2008). As the kernel of $\boldsymbol{L}$ is spanned by constant-valued vectors, this version of the Dirichlet energy is zero when $\boldsymbol{X}^{(k)}$ only has non-zero components belonging to constant-valued vectors. This experiment is based on Rusch, B. Chamberlain, et al. (2022).

Metrics to Quantify Over-Smoothing D. Chen et al. (2020) propose to quantify over-smoothing by computing the average cosine distance between all pairs of nodes, which they call mean average distance (MAD). They consider smoothing as a natural process in GNNs, with smoothing between nodes belonging to the same class as desired. For nodes of different classes, this smoothing adds noise and is undesired. Accordingly, they compute the MAD separately for nodes with the same label and nodes with different labels. The difference between these values is referred to as MADgap.

Liu et al. (2020) study the over-smoothing phenomenon by introducing a novel metric called SMV that computes the average Euclidean distance between a pair of nodes. They consider normalized node representations, ensuring that the magnitude of the representations does not affect the metric.

The Dirichlet energy or Dirichlet sum as a metric for theoretically studying over-smoothing was proposed by Cai et al. (2020). In following studies, the Dirichlet energy was also used to empirically identify methods suffering from the over-smoothing phenomenon (Zhou et al. 2021). Rusch, Bronstein, et al. (2023) generalize the Dirichlet energy to node-similarity measures that aim to quantify the similarity of node representations to a constant state. The Dirichlet energy

$$E(\boldsymbol{X}) = \operatorname{tr}\left(\boldsymbol{X}^\top \boldsymbol{L}\boldsymbol{X}\right) \tag{3.29}$$



(a) Random filter. **(b)** Random filter. **(c)** Random filter.

Figure 3.5: Coefficient magnitudes $|F(\boldsymbol{\theta})|$ of three random filters $\boldsymbol{\theta}$ in graph Fourier space as used for the spectral graph convolution. The KarateClub dataset was used to construct $\boldsymbol{L}_{\text{sym}}$.

defined with the unnormalized graph Laplacian $\boldsymbol{L}$ is one metric that satisfies their criteria.

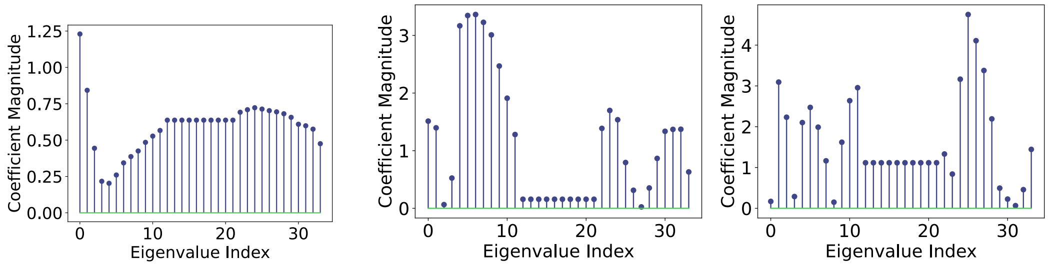
3.2.2 Over-Correlation

One connected phenomenon that our study also connects to is over-correlation. Jin et al. (2022) found the pairwise correlation of feature channels to increase with increased depth. They observe this phenomenon in several message-passing methods by measuring the Pearson correlation coefficient between feature channels. They find that over-smoothing implies over-correlation, but the converse does not hold. As feature channels may be perfectly correlated but contain non-smooth values, over-correlation can arise independently of over-smoothing. They accordingly distinguish that over-smoothing corresponds to a node-wise smoothness, while over-correlation corresponds to a channel-wise similarity. They conclude that additional feature channels are redundant and provide no additional information.

X. Guo et al. (2023) study the effective rank of representations of GNNs and transformers. They find that most singular values of node representations are closer to zero after a few iterations. Similarly to over-correlation, they find this phenomenon to occur even when representations are not similar, as measured by the cosine similarity. They refer to this phenomenon as a more general form of over-smoothing, which they call dimensional collapse. The phenomenon of dimensional collapse was similarly observed for contrastive learning methods (Gao et al. 2019; Jing et al. 2022). The theoretical details of over-correlation and dimensional collapse, as well as their relationship to over-smoothing, are not well understood.

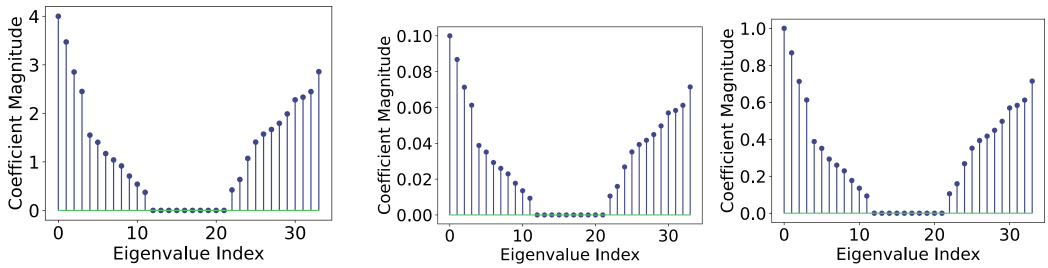
3.3 Over-Smoothing in Graph Fourier Space

Our first analysis to improve our understanding of the over-smoothing phenomenon considers the filters of different graph convolutions. This section closely follows our previ-



(a) $K = 2$. **(b)** $K = 8$. **(c)** $K = 16$.

Figure 3.6: Coefficient magnitudes $|F(\boldsymbol{\theta})|$ of Chebyshev polynomials $F(\boldsymbol{\theta}) = \sum_{k=0}^{K} w_k T_k(\boldsymbol{\lambda})$ as filters in graph Fourier space with three different degrees K. The KarateClub dataset was used to construct $\boldsymbol{L}_{\text{sym}}$.



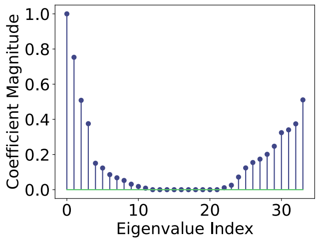
(a) $w = 4$. **(b)** $w = 0.1$. **(c)** $w = -1$.

Figure 3.7: Coefficient magnitudes $|F(\boldsymbol{\theta})|$ of three filters $F(\boldsymbol{\theta}) = w\boldsymbol{\lambda}$ as proposed for the GCN. The KarateClub dataset was used to construct $\boldsymbol{L}_{\text{sym}}$.

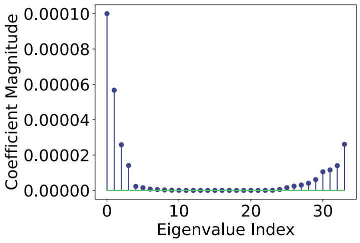
ously published work presented in (Roth and Liebig 2025). As described in Section 2.3.2, the graph convolution is defined in the graph Fourier space

$$F(\boldsymbol{\theta} * \boldsymbol{x}) = F(\boldsymbol{\theta}) \odot F(\boldsymbol{x}) \tag{3.30}$$

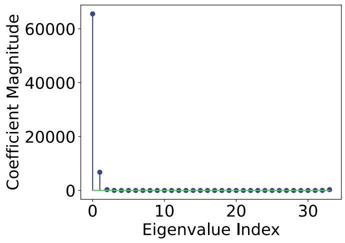
where $\boldsymbol{x} \in \mathbb{R}^n$ are the node attributes or the graph signal, $\boldsymbol{\theta} \in \mathbb{R}^n$ is a filter, and $\odot$ is the element-wise product. The Fourier transform F is defined as the graph Fourier transform given as $F(\boldsymbol{z}) = \boldsymbol{U}^\top \boldsymbol{z}$ for any vector $\boldsymbol{z} \in \mathbb{R}^n$, where $\boldsymbol{U}$ is the matrix of eigenvectors of the graph Laplacian $\boldsymbol{L}_{\text{sym}}$ (see Section 2.2.2). Here, we visualize several exemplary filters to build intuition for the over-smoothing phenomenon and to support our theoretical analysis in the following sections. Each coefficient $F(\boldsymbol{\theta})_i \in \mathbb{R}$ determines the extent to which the component $F(\boldsymbol{x})_i$, associated with the i-th Fourier basis vector, is amplified or dampened. A small magnitude of the coefficient indicates that the associated component gets dampened, while a large magnitude indicates an amplification of the component. Ideally, a convolution can amplify and dampen arbitrary components, depending on the choice of the filter $\boldsymbol{\theta}$.



(a) Combining $k = 2$ GCN filters.



(b) Combining $k = 4$ GCN filters.



(c) Combining $k = 16$ GCN filters.

Figure 3.8: Coefficient magnitudes $|F(\boldsymbol{\theta})|$ of three filters $F(\boldsymbol{\theta}) = w_k\boldsymbol{\lambda}\odot\cdots\odot w_1\boldsymbol{\lambda}$ with different numbers of iterations as proposed for the GCN. The KarateClub dataset was used to construct $\boldsymbol{L}_{\text{sym}}$.

3.3.1 Graph Convolutions as Spectral Filters

The spectral graph convolution allows for any filter $\boldsymbol{\theta}$, as detailed in Section 2.3.2. Thus, any filter $F(\boldsymbol{\theta}) = \boldsymbol{U}^\top\boldsymbol{\theta}$ in the graph Fourier space can be represented. We visualize random choices for $\boldsymbol{\theta}$ in Figure 3.5, visually confirming that this spectral graph convolution can amplify and dampen arbitrary components of the signal $\boldsymbol{x}$.

3.3.2 Polynomial Approximations as Spectral Filters

For a more efficient runtime, polynomial approximations of $F(\boldsymbol{\theta})$ were proposed (Hammond et al. 2011; M. He et al. 2021). These typically utilize the eigenvalues $\boldsymbol{\lambda} \in \mathbb{R}^n$ of the graph Laplacian as a basis. We consider approximations using Chebyshev polynomials $F(\boldsymbol{\theta}) = \sum_{k=0}^{K} w_k T_k(\boldsymbol{\lambda})$ where T_k is the k-th order Chebyshev polynomial with $T_0(\boldsymbol{\lambda}) = \mathbf{1}$, $T_1(\boldsymbol{\lambda}) = \boldsymbol{\lambda}$, and $T_k(\boldsymbol{\lambda}) = 2\cdot\boldsymbol{\lambda}\odot T_{k-1}(\boldsymbol{\lambda}) - T_{k-2}(\boldsymbol{\lambda})$ for $k > 1$ (Hammond et al. 2011). We visualize such polynomial filters of varying degrees in Figure 3.6. We observe a smooth polynomial structure of the filter coefficients, with similar eigenvalues being assigned similar values.

3.3.3 Message-Passing as Spectral Filters

The GCN (Kipf et al. 2017) was derived as a first-order approximation of polynomial graph filters, so that the state of each node is only affected by its direct neighbors. It can be expressed as a graph convolution with a filter in the graph Fourier domain given by $F(\boldsymbol{\theta}) = w\boldsymbol{\lambda}$, where $w \in \mathbb{R}$ is its single parameter. Note that the GCN adds self-loops for computing $\boldsymbol{L}_{\text{sym}}$, which we ignore here for clarity. We visualize filters $F(\boldsymbol{\theta}) = w\boldsymbol{\lambda}$ in graph Fourier space for three values of $w \in \mathbb{R}$ in Figure 3.7. As w uniformly scales all filter coefficients by the same value, the relative magnitudes remain constant for any choice of $w \in \mathbb{R}$. Figure 3.7 also visually highlights that the first coefficient $(w\boldsymbol{\lambda})_1$ always has the largest magnitude among all filter coefficients, for any non-zero $w \in \mathbb{R}$.

Thus, the corresponding smooth component gets amplified maximally for any parameter choice.

This already provides intuition for the over-smoothing phenomenon: under the first-order approximation used by the GCN, any expressible filter $F(\boldsymbol{\theta})$ amplifies the smoothest component the most, as any choice of the parameter w scales all filter coefficients uniformly. When applying the GCN, it is not possible to amplify any component more than the component corresponding to $\boldsymbol{U}_{:,1}$ for any choice of parameter w. Thus, every such filter amplifies the smoothest frequency component the most, while relatively suppressing all other frequency components.

Shared Component Amplification

The provided visualizations considered the single-input single-output (SISO) case, where $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{\theta} * \boldsymbol{x} \in \mathbb{R}^n$ are vectors, i.e., each node has a single feature. The above methods are typically applied in the multi-input multi-output (MIMO) case, where the graph signal $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ has d feature channels for each node. Based on Bruna et al. (2014), the graph convolution is applied in the MIMO case by applying a separate graph convolution with independent filters $\boldsymbol{\theta}_{(p,q)} \in \mathbb{R}^n$ for every combination of input feature channel $i \in [d]$ and output feature channel $q \in [c]$, i.e., the resulting state is obtained as

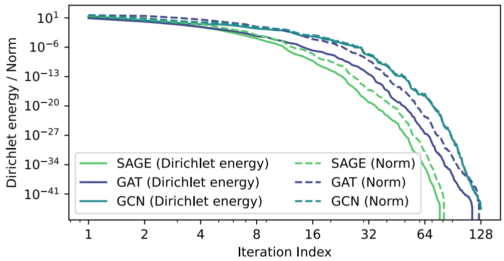
$$F\left(\boldsymbol{X}'_{:,q}\right) = \sum_{p \in [d]} F\left(\boldsymbol{\theta}_{(p,q)}\right) \odot F\left(\boldsymbol{X}_{:,p}\right) \tag{3.31}$$

with $\boldsymbol{X}' \in \mathbb{R}^{n \times c}$. The GCN also utilizes this extension to the MIMO case (Kipf et al. 2017), for which we have $F\left(\boldsymbol{\theta}_{(p,q)}\right) = w_{(p,q)}\boldsymbol{\lambda}$ with $w_{(p,q)} \in \mathbb{R}$ for every pair of input channel p and output channel q. The parameters $w_{(p,q)}$ can also be arranged into a matrix $\boldsymbol{W} \in \mathbb{R}^{d \times c}$, where $W_{p,q} = e_{(p,q)}$. Equation (3.31) can be equivalently stated for the GCN in the typical notation as

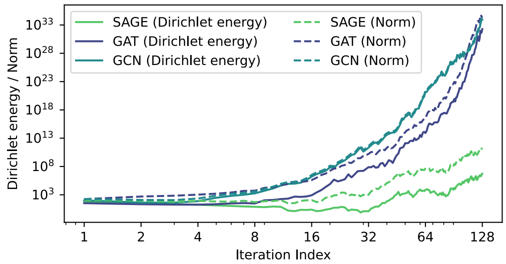
$$\boldsymbol{X}' = \boldsymbol{A}_{\text{sym}}\boldsymbol{X}\boldsymbol{W} \, . \tag{3.32}$$

We recall that for every choice of parameters $W_{p,q}$, each $F\left(\boldsymbol{\theta}_{(p,q)}\right)$ amplifies the frequency components in the same way up to a scaling factor that is uniform across all components. Thus, the relative amplification of components is shared across all pairs of input and output feature channels. We refer to this as shared component amplification (SCA). The resulting representations are inherently constrained, and the gains of additional output channels are limited. We will study this phenomenon more deeply theoretically in the following sections and provide a formal definition in Definition 3.1.

Importantly, SCA occurs regardless of the specific basis $\boldsymbol{\lambda}$ used. Even when a different basis $\hat{\boldsymbol{\lambda}}$ is chosen, the parameters $W_{p,q}$ lead to filters with the same relative amplification of components across all channel combinations. The component corresponding to the maximal absolute value of $\hat{\boldsymbol{\lambda}}$ is maximally amplified for every input and output combination of feature channels. Instead, it is desirable to allow for different components in the input signal to be amplified across distinct output channels.



(a) Dirichlet energy converges to zero.



(b) Dirichlet energy does not converge.

Figure 3.9: Comparison of the Dirichlet energy $E(\boldsymbol{X}^{(k)})$ defined using the random walk Laplacian $\boldsymbol{L}_{\mathrm{rw}}$ and the norm of the representations $\left\|\boldsymbol{X}^{(k)}\right\|_F^2$ on the Cora dataset (McCallum et al. 2000). The node representations $\boldsymbol{X}^{(k)}$ are obtained for 128 iterations of three MPNNs (graph convolutional network (GCN) (Kipf et al. 2017), graph attention network (GAT) (Velickovic et al. 2018), the SAGE convolution (Hamilton et al. 2017)). Following Rusch, Bronstein, et al. (2023), random parameters are used for (a). We additionally present the same figure when all parameters are scaled by a factor of 2 in (b).

Component Dominance

SCA highlights a limitation in the approximation of a single graph convolution $\boldsymbol{\theta} * \boldsymbol{x}$ used in the GCN. For the GCN as a localized approximation, the resulting state for each node takes information only from its direct neighbors into account. The GCN is typically applied in an iterative fashion. Ignoring the non-linear activation functions, this has form

$$F\left(\boldsymbol{\theta}_k\right) \odot \cdots \odot F\left(\boldsymbol{\theta}_1\right) \odot F(\boldsymbol{x}) = w_k \boldsymbol{\lambda} \odot \cdots \odot w_1 \boldsymbol{\lambda} \odot F(\boldsymbol{x}) \,, \tag{3.33}$$

where $\boldsymbol{\theta}_i \in \mathbb{R}^n$ is the filter for iteration $i \in [k]$, and $w_i \in \mathbb{R}$ is the corresponding parameter of the GCN. We visualize random such filters $F(\boldsymbol{\theta}) = \left(w_k \cdot \cdots \cdot w_1\right) \boldsymbol{\lambda}^{(k)}$, where $\lambda_i^{(k)} = \lambda_i^k$ for different numbers of layers k in Figure 3.8. We observe that with increased depth, all components except the first are increasingly suppressed, as their corresponding eigenvalues λ_i^k approach zero relative to λ_1^k.

As in the single iteration case, the scalar $(w_k \cdot \ldots \cdot w_1) \in \mathbb{R}$ applies a uniform scaling of each coefficient λ_i^k. As such, the maximal coefficient λ_1 increasingly dominates the other coefficients with an increasing number of iterations k, independent of the chosen parameters $w_1, \ldots, w_k$. We refer to the phenomenon of a single component being dominantly amplified as component dominance, which we will formally define in Definition 3.2. These insights provide an initial intuition for the properties of the GCN limiting their performance and the theoretical analysis presented in the following section.

3.4 Over-Smoothing and Vanishing Norms

Several conflicting definitions and views for over-smoothing currently coexist, as detailed in Section 3.2. Several studies prove that node representations converge to a multiple of the degree-proportional dominant eigenvector $\boldsymbol{D}^{1/2}\mathbf{1}$ of $\boldsymbol{A}_{\text{sym}}$ when repeatedly applying GCN iterations (Oono et al. 2020; Cai et al. 2020; Zhou et al. 2021; Giovanni et al. 2023). This can be shown using the Dirichlet energy $E(\boldsymbol{X}) = \text{tr}\left(\boldsymbol{X}^\top \boldsymbol{L}_{\text{sym}}\boldsymbol{X}\right)$, as the degree-proportional state corresponds to the null space of $\boldsymbol{L}_{\text{sym}}$, i.e., $\boldsymbol{L}_{\text{sym}}\boldsymbol{D}^{1/2}\mathbf{1} = \mathbf{0}$. Others have observed an exponential convergence to a constant state (Rusch, B. Chamberlain, et al. 2022; Rusch, B. P. Chamberlain, et al. 2023; Rusch, Bronstein, et al. 2023; Wu et al. 2023), for example, using the Dirichlet energy $E_{\boldsymbol{L}} = \text{tr}\left(\boldsymbol{X}^\top \boldsymbol{L}\boldsymbol{X}\right)$ defined using the unnormalized graph Laplacian $\boldsymbol{L}$, as the constant state is in the null space of $\boldsymbol{L}$, i.e., $\boldsymbol{L}\mathbf{1} = \mathbf{0}$.

Mathematically, the matrix $\boldsymbol{X}$ lies in the intersection of the null spaces of both $\boldsymbol{L}_{\text{sym}}$ and $\boldsymbol{L}$ if and only if $\boldsymbol{X} = \mathbf{0}$, i.e., all entries of $\boldsymbol{X}$ are zero. Thus, the norm of the representations needs to vanish in order for both properties to be satisfied. To verify this claim, we theoretically study the effect of applying a GCN iteration on the norm of the representations. We find that the norm of $\boldsymbol{X}$ is bounded by the largest singular value of $\boldsymbol{W}$, leading to the same bound as shown for the Dirichlet energy ((Oono et al. 2020; Cai et al. 2020)). We formalize this in Proposition 3.3:

Proposition 3.3 (Vanishing norm of node representations). Let $\boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{n\times n}$ be a symmetrically normalized adjacency matrix and $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, $\boldsymbol{W} \in \mathbb{R}^{d\times c}$ arbitrary matrices for some $n, d, c \in \mathbb{N}$. Further, let ϕ be a component-wise activation function with $|\phi(a)| \leq |a|$ for all $a \in \mathbb{R}$. Then,

$$\left\|\phi\left(\boldsymbol{A}_{\text{sym}}\boldsymbol{X}\boldsymbol{W}\right)\right\|_F \leq \sigma_1 \|\boldsymbol{X}\|_F \,, \tag{3.34}$$

where σ_1 is the largest singular value of $\boldsymbol{W}$.

Proof. By definition of the Frobenius norm, we have

$$\left\|\phi\left(\boldsymbol{A}_{\text{sym}}\boldsymbol{X}\boldsymbol{W}\right)\right\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{c}\phi\left(\boldsymbol{A}_{\text{sym}}\boldsymbol{X}\boldsymbol{W}\right)_{i,j}^2}\,. \tag{3.35}$$

Using the property $|\phi(a)| \leq |a|$ for all $a \in \mathbb{R}$ of the activation function ϕ, we obtain

$$\begin{aligned}\left\|\phi\left(\boldsymbol{A}_{\text{sym}}\boldsymbol{X}\boldsymbol{W}\right)\right\|_F &\leq \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{c}\left(\boldsymbol{A}_{\text{sym}}\boldsymbol{X}\boldsymbol{W}\right)_{i,j}^2} \\ &= \left\|\boldsymbol{A}_{\text{sym}}\boldsymbol{X}\boldsymbol{W}\right\|_F,\end{aligned} \tag{3.36}$$

which is the Frobenius norm, ignoring the non-linear activation function. We now apply the sub-multiplicative property of the Frobenius norm with the spectral norm, yielding

$$\left\|\boldsymbol{A}_{\text{sym}}\boldsymbol{X}\boldsymbol{W}\right\|_F \leq \left\|\boldsymbol{A}_{\text{sym}}\right\|_2 \cdot \|\boldsymbol{X}\|_F \cdot \|\boldsymbol{W}\|_2 \,. \tag{3.37}$$

We then use the fact that the spectral norm of a matrix is equal to its largest singular value, i.e., $\left\|\boldsymbol{A}_{\text{sym}}\right\|_2 = \lambda_1 = 1$ by properties of $\boldsymbol{A}_{\text{sym}}$ and $\|\boldsymbol{W}\|_2 = \sigma_1$. Combining these equalities, we obtain

$$\left\|\phi\left(\boldsymbol{A}_{\text{sym}}\boldsymbol{X}\boldsymbol{W}\right)\right\|_F \leq \sigma_1\|\boldsymbol{X}\|_F\,, \tag{3.38}$$

as claimed. □

Proposition 3.3 corresponds to applying a single GCN iteration with

$$\boldsymbol{X}^{(k)} = \phi\left(\boldsymbol{A}_{\text{sym}}\boldsymbol{X}^{(k-1)}\boldsymbol{W}^{(k)}\right)\,. \tag{3.39}$$

When repeatedly applying such GCN iterations, all entries of the state converge to zero when $\prod_{k=1}^{\infty}\sigma_1^{\boldsymbol{W}^{(k)}} = 0$ with $\sigma_1^{\boldsymbol{W}^{(k)}}$ being the largest singular value of $\boldsymbol{W}^{(k)}$, i.e., we have $\lim_{k\to\infty}\boldsymbol{X}^{(k)} = 0$. As the zero matrix is a special case of both the constant vector $\mathbf{1}$ and the degree-proportional vector $\boldsymbol{D}^{1/2}\mathbf{1}$, the Dirichlet energy $E(\boldsymbol{X})$ defined using $\boldsymbol{L}_{\text{sym}}$ and the Dirichlet energy $E_{\boldsymbol{L}}(\boldsymbol{X})$ defined using $\boldsymbol{L}$ both approach zero as $k\to\infty$. This explains the previous seemingly contradictory findings.

We visualize both the Frobenius norm $\left\|\boldsymbol{X}^{(k)}\right\|_F$ and the Dirichlet energy $E_{\boldsymbol{L}}\left(\boldsymbol{X}^{(k)}\right)$ for node representations according to Equation (3.39) in Figure 3.9. We observe a close similarity between both metrics. This is not only the case when the norm converges to zero, but also when the norm does not converge to zero. Thus, the norm of the node representations overshadows potential insights of the Dirichlet energy. As such, observing the Dirichlet energy to converge to zero seems to be a consequence of the norm converging to zero, and not of the representations converging to a degree-dependent or constant state. This calls into question the potential insights gained from the Dirichlet energy in this form.

This vanishing norm is a critical concern when applying many message-passing iterations, which can lead to vanishing gradients (Bengio et al. 1994; Hochreiter et al. 1997; Pascanu et al. 2013) and hinder optimization. As previously shown, the over-smoothing phenomenon arises independently of the scale of node representations (Liu et al. 2020; D. Chen et al. 2020; Giovanni et al. 2023; Maskey et al. 2023). Instead, we argue to separate the vanishing norm from over-smoothing and study both phenomena independently. This allows a more targeted analysis of the reasons and the effects of each property. Separating them also allows for better communication. The vanishing norm always implies over-smoothing under this definition, since $E(\mathbf{0}) = 0$. This would imply over-smoothing even for trivial cases, such as $\boldsymbol{X}^{(k)} = a\boldsymbol{X}^{(k-1)}$ for $|a| < 1$. This suffers from the vanishing norm problem and over-smoothing in this definition, but we argue that it distracts from the underlying problem of over-smoothing. Thus, when the representations converge to zero, it is a misleading special case. Several theoretical studies base their findings about over-smoothing on assumptions about the singular values of the feature transformations (Oono et al. 2020; Cai et al. 2020; Zhou et al. 2021; Wu et al. 2023), which may oversimplify certain cases, as these primarily concern the magnitude of the features, rather than the smoothing property.

Other existing metrics, such as SMV (Liu et al. 2020) and MAD (D. Chen et al. 2020) (see Section 3.2.1), similarly argue to quantify the similarity of node representations independently of the feature magnitude. These incorporate feature normalization to consider over-smoothing independently of the norm. However, these metrics are not used for theoretical studies. More recent theoretical studies on over-smoothing consider the Dirichlet energy $E\left(\frac{\boldsymbol{X}^{(k)}}{\|\boldsymbol{X}^{(k)}\|_F}\right)$ of the normalized node representations, which can be considered as a study of over-smoothing while ignoring the effects of the norm (Giovanni et al. 2023; Maskey et al. 2023). We further argue that the Dirichlet energy alone might not provide sufficient insight into the underlying phenomenon.

This observation aligns with our initial study on convolutional filters in the graph Fourier domain in Section 3.3.3. We observed that all components are uniformly scaled, which may overshadow the amplification of the smooth component.

3.5 A Theoretical Study on the Effect of Message-Passing on Representations

Based on Section 3.2.2, we have developed some intuition about the effects of applying iterations of the GCN and provided some initial insights into the phenomena of shared component amplification (SCA) and component dominance (CD). We have also observed that the norm of node representations has led to some conflicting findings and definitions regarding over-smoothing. Here, we conduct an in-depth study on the theoretical properties of the GCN, which we will later generalize. We consider linearized message-passing functions of the form

$$\boldsymbol{X}^{(k)} = \tilde{\boldsymbol{A}}\boldsymbol{X}^{(k-1)}\boldsymbol{W}^{(k)} \tag{3.40}$$

where $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n\times n}$, $\boldsymbol{X}^{(k)} \in \mathbb{R}^{n\times d}$, and $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d\times d}$ for all $k \in \mathbb{N}$ and some $d \in \mathbb{N}$. This corresponds to the equivalent node-wise update function

$$\boldsymbol{X}^{(k)}_{i,:} = \sum_{j\in\mathcal{N}_i} A_{i,j}\boldsymbol{X}^{(k-1)}_{j,:}\boldsymbol{W}^{(k)}\,. \tag{3.41}$$

Theoretical insights are currently only available for the case when $\tilde{\boldsymbol{A}} := \boldsymbol{A}_{\text{sym}}$ and the feature transformations $\boldsymbol{W}^{(k)} = \boldsymbol{W}$ are shared for every $k \in \mathbb{N}$ with the shared matrix $\boldsymbol{W}$ also being symmetric (Giovanni et al. 2023). Their results only apply to the limit case $\boldsymbol{X}^{(l)}$ with infinitely many iterations l. We study the effect of such a message-passing step for any, not necessarily identical, matrices $\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(l)}$ of suitable shape, both for a single iteration and the repeated application. We initially consider the case $\tilde{\boldsymbol{A}} := \boldsymbol{A}_{\text{sym}}$ and then further generalize our theoretical study to allow for any aggregation matrix $\tilde{\boldsymbol{A}}$ instead of the symmetrically normalized adjacency matrix $\boldsymbol{A}_{\text{sym}}$. We summarize this connection between our findings and the currently known case in Figure 3.10. We summarize our findings and provide comprehensive definitions in Section 3.6. This analysis mostly follows one of our previous publications (Roth and Liebig 2023).

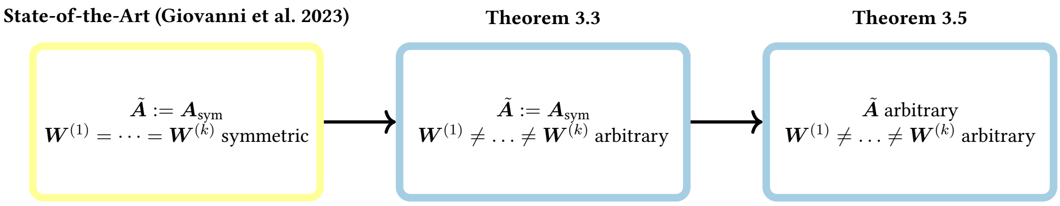


Figure 3.10: Limit behavior of $\boldsymbol{X}^{(k)}$, where $\boldsymbol{X}^{(k)} = \tilde{\boldsymbol{A}}\boldsymbol{X}^{(k-1)}\boldsymbol{W}^{(k)}$, under various choices for $\tilde{\boldsymbol{A}}$ and $\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(k)}$ from existing state-of-the-art, and our proposed extensions.

3.5.1 The Effect of GCN Iterations

We now assume the case $\tilde{\boldsymbol{A}} := \boldsymbol{A}_{\text{sym}}$ in Equation (3.40) for an undirected and non-bipartite graph, as used for the GCN (Kipf et al. 2017). Instead of focusing on the Dirichlet energy, we study the effect of the GCN on the node representations in general. To ease our theoretical study, we utilize the vectorized form of a single iteration of the GCN from Equation (3.40), resulting in

$$\operatorname{vec}\left(\boldsymbol{A}_{\text{sym}}\boldsymbol{X}\boldsymbol{W}\right) = \left(\boldsymbol{W}^{\top} \otimes \boldsymbol{A}_{\text{sym}}\right) \operatorname{vec}(\boldsymbol{X}) = \boldsymbol{T} \operatorname{vec}(\boldsymbol{X}) \tag{3.42}$$

with $\boldsymbol{T} = \left(\boldsymbol{W}^{\top} \otimes \boldsymbol{A}_{\text{sym}}\right) \in \mathbb{R}^{nd \times nd}$ applied before any non-linear activation function. The Kronecker product $\otimes$ is a standard operation that is frequently used for theoretical studies of GNNs (Gu et al. 2020; Roth and Liebig 2022b; Giovanni et al. 2023; Maskey et al. 2023). We will introduce its properties as required. The first important property of the Kronecker product used above is $\operatorname{vec}(\boldsymbol{A}\boldsymbol{B}\boldsymbol{C}) = \left(\boldsymbol{C}^{\top} \otimes \boldsymbol{A}\right) \operatorname{vec}(\boldsymbol{B})$ for any matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ of suitable shape (Horn et al. 1991). As we allow for arbitrary matrices $\boldsymbol{W}$, we will drop the transpose of $\boldsymbol{W}$ in the notation for conciseness. Next, we represent the node representations in the graph Fourier basis (see Section 2.2.2) $\boldsymbol{U}$ using the property $\boldsymbol{U}^{\top}\boldsymbol{U} = \boldsymbol{I}_n$ as follows:

$$\begin{aligned} \operatorname{vec}(\boldsymbol{X}) &= \operatorname{vec}(\boldsymbol{U}\boldsymbol{B}) \\ &= (\boldsymbol{I}_d \otimes \boldsymbol{U}) \operatorname{vec}(\boldsymbol{B}) \end{aligned} \tag{3.43}$$

where $\boldsymbol{B} = \boldsymbol{U}^{\top}\boldsymbol{X} \in \mathbb{R}^{n \times d}$ contains the graph Fourier transformed node representations or components. Each entry $B_{i,j}$ represents the coefficient or component of the i-th graph Fourier basis vector of feature channel j. We express the graph Fourier decomposition of $\boldsymbol{X}$ as

$$\begin{aligned} \operatorname{vec}(\boldsymbol{X}) &= \sum_{i=1}^{n} (\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}) \operatorname{vec}\left(\boldsymbol{B}_{i,:}\right) \\ &= \sum_{i=1}^{n} \boldsymbol{S}_{(i)} \operatorname{vec}\left(\boldsymbol{B}_{i,:}\right) , \end{aligned} \tag{3.44}$$

where $\boldsymbol{S}_{(i)} = (\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}) \in \mathbb{R}^{nd \times d}$ and $\operatorname{vec}\left(\boldsymbol{B}_{i,:}\right) \in \mathbb{R}^{d}$ is the vector containing the coefficients of the i-th graph Fourier basis vector across all feature channels. Combining this reformulation of the state $\boldsymbol{X}$ with the vectorized GCN iteration (Equation (3.42)), we consider the effect of a GCN iteration

$$\operatorname{vec}\left(\boldsymbol{A}_{\text{sym}}\boldsymbol{X}\boldsymbol{W}\right) = \sum_{i=1}^{n} \boldsymbol{T}\boldsymbol{S}_{(i)} \operatorname{vec}\left(\boldsymbol{B}_{i,:}\right) \tag{3.45}$$

on the different graph Fourier components. For each graph Fourier basis vector, we further construct disjoint subspaces

$$\mathcal{Q}_i = \operatorname{span}\left\{ \boldsymbol{S}_{(i)}\boldsymbol{v} \mid \boldsymbol{v} \in \mathbb{R}^d \right\} . \tag{3.46}$$

Every element $\operatorname{vec}(\boldsymbol{Z}) \in \mathcal{Q}_i$ is the vectorized form of a rank-one matrix $\boldsymbol{Z} = \boldsymbol{U}_{:,i}\boldsymbol{c}^\top \in \mathbb{R}^{n\times d}$ for some $\boldsymbol{c} \in \mathbb{R}^d$. $\boldsymbol{Z}$ only contains non-zero components corresponding to graph Fourier basis vector $\boldsymbol{U}_{:,i}$. Thus, each column of $\boldsymbol{Z}$ is a scalar multiple of the corresponding graph Fourier basis vector $\boldsymbol{U}_{:,i}$. We find the first key property of GCN iterations to exploit this decomposition. Each subspace corresponding to a graph Fourier basis vector is invariant under applications of any GCN iteration $\boldsymbol{T}$:

Lemma 3.1 (Graph Fourier Components are invariant to GCN iterations). Let $\boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{n\times n}$ be a symmetrically normalized adjacency matrix with an orthogonal matrix of eigenvectors $\boldsymbol{U}$, and let $\boldsymbol{W} \in \mathbb{R}^{d\times d}$ be any matrix. Consider the subspace

$$\mathcal{Q}_i = \operatorname{span}\left\{ \left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}\right)\boldsymbol{v} \mid \boldsymbol{v} \in \mathbb{R}^d \right\} .$$

Then, $\mathcal{Q}_i$ is invariant under $\boldsymbol{T} = \boldsymbol{W} \otimes \boldsymbol{A}_{\text{sym}}$, i.e.,

$$\forall \boldsymbol{z} \in \mathcal{Q}_i \colon \boldsymbol{T}\boldsymbol{z} \in \mathcal{Q}_i . \tag{3.47}$$

Proof. Any $\boldsymbol{z} \in \mathcal{Q}_i$ can be expressed as a linear combination $\boldsymbol{z} = \left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}\right)\boldsymbol{b}$ of basis vectors $\left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}\right)$ of $\mathcal{Q}_i$, where $\boldsymbol{b} \in \mathbb{R}^{nd}$. By definition of $\boldsymbol{T} = \boldsymbol{W} \otimes \boldsymbol{A}_{\text{sym}}$, we have

$$\boldsymbol{T}\boldsymbol{z} = \left(\boldsymbol{W} \otimes \boldsymbol{A}_{\text{sym}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}\right)\boldsymbol{b} . \tag{3.48}$$

Following the mixed-product property $(\boldsymbol{A} \otimes \boldsymbol{B}) \cdot (\boldsymbol{C} \otimes \boldsymbol{D}) = (\boldsymbol{A} \cdot \boldsymbol{C}) \otimes (\boldsymbol{B} \cdot \boldsymbol{D})$ of the Kronecker product and the matrix product for matrices of suitable shape and $\boldsymbol{A}_{\text{sym}}\boldsymbol{U}_{:,i} = \lambda_i \boldsymbol{U}_{:,i}$, we have

$$\left(\boldsymbol{W} \otimes \boldsymbol{A}_{\text{sym}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}\right)\boldsymbol{b} = \left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}\right)\left(\boldsymbol{W} \otimes \lambda_i \boldsymbol{I}_n\right)\boldsymbol{b} . \tag{3.49}$$

By defining $\boldsymbol{b}' = \left(\boldsymbol{W} \otimes \lambda_i \boldsymbol{I}_n\right)\boldsymbol{b}$, this further simplifies to

$$\boldsymbol{T}\boldsymbol{z} = \left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}\right)\boldsymbol{b}' . \tag{3.50}$$

As $\boldsymbol{T}\boldsymbol{z}$ is a linear combination of the basis vectors from subspace $\mathcal{Q}_i$, this concludes the proof. □

This proof exploits two central properties of the constructed subspaces. First, as the graph Fourier basis vectors are also eigenvectors of $\boldsymbol{A}_{\text{sym}}$, applying $\boldsymbol{A}_{\text{sym}}$ reduces to scaling each graph Fourier component by the corresponding eigenvalue. Second, applying the feature transformation $\boldsymbol{W}$ cannot transform the given graph Fourier components into other graph Fourier components. This discovery is central to our subsequent investigation, as it enables us to study the effect of a GCN iteration separately for each subspace and compare their effects.

A Single GCN Iteration

Given these invariant subspaces $\mathcal{Q}_i$, we now study the effect of GCN iterations on each graph Fourier component independently. Precisely, we study the effect of a vectorized GCN iteration $\boldsymbol{W} \otimes \boldsymbol{A}_{\text{sym}}$ on each set of graph Fourier basis vectors $\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}$ of subspace $\mathcal{Q}_i$, i.e., we consider $\left(\boldsymbol{W} \otimes \boldsymbol{A}_{\text{sym}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}\right)$ for all $i \in [n]$. We find $\boldsymbol{A}_{\text{sym}}$ to perform a scaling of the basis vectors $\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}$ of each subspace $\mathcal{Q}_i$ by the corresponding eigenvalue λ_i of $\boldsymbol{A}_{\text{sym}}$, i.e.,

$$\left(\boldsymbol{I}_d \otimes \boldsymbol{A}_{\text{sym}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}\right) = \lambda_i \left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}\right) . \tag{3.51}$$

The feature transformation $\boldsymbol{W}$ performs an equivalent transformation to all sets of basis vectors $\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}$. While this can change the norm $\left\|\left(\boldsymbol{W} \otimes \boldsymbol{I}_n\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}\right)\right\|_F$ arbitrarily, the norm of all subspaces equally changes, i.e., we have

$$\frac{\left\|\left(\boldsymbol{W} \otimes \boldsymbol{I}_n\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}\right)\right\|_F}{\left\|\left(\boldsymbol{W} \otimes \boldsymbol{I}_n\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,j}\right)\right\|_F} = 1 \tag{3.52}$$

for all $i, j \in [n]$. Thus, the feature transformation cannot amplify components for one graph Fourier basis vector more than for another. Applying both $\boldsymbol{A}_{\text{sym}}$ and $\boldsymbol{W}$ then scales the basis vectors of each subspace $\mathcal{Q}_i$ only relative to the corresponding eigenvalue λ_i. We formalize this finding in the following theorem:

Theorem 3.2 (Shared Component Amplification for the GCN). Let $\boldsymbol{T} = \boldsymbol{W} \otimes \boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{nd \times nd}$, where $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ is any matrix and $\boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{n \times n}$ is the symmetrically normalized adjacency matrix with orthonormal eigenvectors $\boldsymbol{U} \in \mathbb{R}^{n \times n}$. Let $\boldsymbol{S}_{(i)} = \boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}$ with corresponding eigenvalue λ_i of $\boldsymbol{A}_{\text{sym}}$ for all $i \in [n]$. Then, for all $i, j \in [n]$,

$$\frac{\left\|\boldsymbol{T}\boldsymbol{S}_{(i)}\right\|_F}{\left\|\boldsymbol{T}\boldsymbol{S}_{(j)}\right\|_F} = \frac{|\lambda_i|}{|\lambda_j|} . \tag{3.53}$$

Proof. We start by expanding all matrices by their definitions:

$$\frac{\left\|\boldsymbol{T}\boldsymbol{S}_{(i)}\right\|_F}{\left\|\boldsymbol{T}\boldsymbol{S}_{(j)}\right\|_F} = \frac{\left\|\left(\boldsymbol{W} \otimes \boldsymbol{A}_{\text{sym}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}\right)\right\|_F}{\left\|\left(\boldsymbol{W} \otimes \boldsymbol{A}_{\text{sym}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,j}\right)\right\|_F} . \tag{3.54}$$

Using the mixed-product property of the Kronecker product and the matrix product, we have

$$\frac{\left\|\left(\boldsymbol{W} \otimes \boldsymbol{A}_{\text{sym}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}\right)\right\|_F}{\left\|\left(\boldsymbol{W} \otimes \boldsymbol{A}_{\text{sym}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,j}\right)\right\|_F} = \frac{\left\|\left(\boldsymbol{W} \otimes \boldsymbol{A}_{\text{sym}}\boldsymbol{U}_{:,i}\right)\right\|_F}{\left\|\left(\boldsymbol{W} \otimes \boldsymbol{A}_{\text{sym}}\boldsymbol{U}_{:,j}\right)\right\|_F} . \tag{3.55}$$

As $\boldsymbol{U}_{:,i}$ is an eigenvector of $\boldsymbol{A}_{\text{sym}}$ with eigenvalue λ_i, we have

$$\frac{\left\|\left(\boldsymbol{W} \otimes \boldsymbol{A}_{\text{sym}}\boldsymbol{U}_{:,i}\right)\right\|_F}{\left\|\left(\boldsymbol{W} \otimes \boldsymbol{A}_{\text{sym}}\boldsymbol{U}_{:,j}\right)\right\|_F} = \frac{\|(\boldsymbol{W} \otimes \lambda_i\boldsymbol{U}_{:,i})\|_F}{\|(\boldsymbol{W} \otimes \lambda_j\boldsymbol{U}_{:,j})\|_F}\,. \tag{3.56}$$

By sub-multiplicativity of the norm of the Kronecker product, this simplifies to

$$\frac{\|\boldsymbol{W} \otimes \lambda_i\boldsymbol{U}_{:,i}\|_F}{\|\boldsymbol{W} \otimes \lambda_j\boldsymbol{U}_{:,j}\|_F} = \frac{\|\boldsymbol{W}\|_F \cdot \|\lambda_i\boldsymbol{U}_{:,i}\|_F}{\|\boldsymbol{W}\|_F \cdot \|\lambda_j\boldsymbol{U}_{:,j}\|_F}\,. \tag{3.57}$$

The norm $\|\boldsymbol{W}\|_F$ cancels, and with the absolute homogeneity property of the norm, we have

$$\frac{\|\boldsymbol{W}\|_F \cdot \|\lambda_i\boldsymbol{U}_{:,i}\|_F}{\|\boldsymbol{W}\|_F \cdot \|\lambda_j\boldsymbol{U}_{:,j}\|_F} = \frac{|\lambda_i| \cdot \|\boldsymbol{U}_{:,i}\|_F}{|\lambda_j| \cdot \|\boldsymbol{U}_{:,j}\|_F}\,. \tag{3.58}$$

As each $\boldsymbol{U}$ is chosen to be an orthonormal matrix, the Frobenius norm of $\boldsymbol{U}_{:,i}$ is one, and we have

$$\frac{|\lambda_i| \cdot \|\boldsymbol{U}_{:,i}\|_F}{|\lambda_j| \cdot \|\boldsymbol{U}_{:,j}\|_F} = \frac{|\lambda_i|}{|\lambda_j|}\,. \tag{3.59}$$

This concludes the proof. □

This property of the GCN can be interpreted in several ways. As the eigenvalues are fixed and given by $\boldsymbol{A}_{\text{sym}}$, the subspaces with larger eigenvalues will be amplified more strongly than those corresponding to eigenvalues with a smaller magnitude for any $\boldsymbol{W}$. As the largest eigenvalue $\lambda_1 = 1$, with corresponding smooth eigenvector $\boldsymbol{U}_{:,1} = \boldsymbol{D}^{1/2}\mathbf{1}$ that is degree-proportional, each iteration relatively amplifies this smooth subspace by this fixed ratio compared to all other subspaces.

In addition to this relative amplification being fixed, it is also shared across all feature channels. As every $\boldsymbol{v} \in \mathcal{Q}_i$ is the vectorized form of a rank-one matrix, the same graph Fourier basis vector is relatively amplified by the same ratio of eigenvalue for every feature channel. Thus, a single GCN iteration relatively amplifies the smooth components across all feature channels by the ratio $\frac{\lambda_1}{\lambda_i}$ compared to all other components. We will define a more general form of this phenomenon in Definition 3.1, and refer to this phenomenon as shared component amplification (SCA). This finding also aligns with our spectral analysis of the filter used within the GCN in Section 3.3.3.

As the ratio depends on the aggregation function $\boldsymbol{A}_{\text{sym}}$ but not on the feature transformation $\boldsymbol{W}$, we cannot construct or learn feature transformations that prevent the amplification of smooth signals for any of the feature channels.

Compared to previous studies that identified bounds on the Dirichlet energy based on singular values of $\boldsymbol{W}$ (Oono et al. 2020; Cai et al. 2020), this equality provides deeper insights. This relative amplification of the message-passing step is independent of any subsequent operations, such as non-linear activation functions.

Repeated GCN Iterations

We have identified that each GCN iteration amplifies the graph Fourier components relatively with a fixed ratio given by the corresponding eigenvalues of $\boldsymbol{A}_{\text{sym}}$. We now consider the case when repeatedly applying GCN iterations

$$\operatorname{vec}\left(\boldsymbol{X}^{(k)}\right) = \operatorname{vec}\left(\boldsymbol{A}_{\text{sym}}\boldsymbol{X}^{(k-1)}\boldsymbol{W}^{(k)}\right) = \left(\left(\boldsymbol{W}^{(k)}\right)^{\top} \otimes \boldsymbol{A}_{\text{sym}}\right)\operatorname{vec}\left(\boldsymbol{X}^{(k-1)}\right) \tag{3.60}$$

with time-inhomogeneous feature transformations $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d\times d}$ for every $k \in \mathbb{N}$. As before, we will omit the transpose for notational simplicity, as we allow any matrices $\boldsymbol{W}^{(k)}$. Although the following analysis only covers the linear case without activation functions, this linear case remains insufficiently explored. The extension to the nonlinear case remains open for future work. The relative amplification of graph Fourier induced subspaces from Theorem 3.2 can be directly extended to this iterated case:

Proposition 3.4 (Component amplification for multiple GCN iterations). Let $\boldsymbol{T}^{(k)} = \boldsymbol{W}^{(k)} \otimes \boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{nd\times nd}$, where $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d\times d}$ is any matrix and $\boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{n\times n}$ is the symmetrically normalized adjacency matrix with orthonormal eigenvectors $\boldsymbol{U} \in \mathbb{R}^{n\times n}$. Let $\boldsymbol{S}_{(i)} = \boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}$ with corresponding eigenvalue λ_i of $\boldsymbol{A}_{\text{sym}}$ for all $i \in [n]$. Then, for all $i, j \in [n]$,

$$\frac{\left\|\boldsymbol{T}^{(l)}\dots\boldsymbol{T}^{(1)}\boldsymbol{S}_{(i)}\right\|_F}{\left\|\boldsymbol{T}^{(l)}\dots\boldsymbol{T}^{(1)}\boldsymbol{S}_{(j)}\right\|_F} = \frac{|\lambda_i^l|}{|\lambda_j^l|} \tag{3.61}$$

Proof. This proof is mostly analogous to the proof of Theorem 3.2. We start by replacing each $\boldsymbol{T}^{(k)}$ and $\boldsymbol{S}_{(i)}$ by their definition:

$$\frac{\left\|\boldsymbol{T}^{(l)}\dots\boldsymbol{T}^{(1)}\boldsymbol{S}_{(i)}\right\|_F}{\left\|\boldsymbol{T}^{(l)}\dots\boldsymbol{T}^{(1)}\boldsymbol{S}_{(j)}\right\|_F} = \frac{\left\|\left(\boldsymbol{W}^{(l)} \otimes \boldsymbol{A}_{\text{sym}}\right)\dots\left(\boldsymbol{W}^{(1)} \otimes \boldsymbol{A}_{\text{sym}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}\right)\right\|_F}{\left\|\left(\boldsymbol{W}^{(l)} \otimes \boldsymbol{A}_{\text{sym}}\right)\dots\left(\boldsymbol{W}^{(1)} \otimes \boldsymbol{A}_{\text{sym}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,j}\right)\right\|_F}. \tag{3.62}$$

By the mixed-product property of the Kronecker product and the matrix product, this simplifies to

$$\frac{\left\|\left(\boldsymbol{W}^{(l)} \otimes \boldsymbol{A}_{\text{sym}}\right)\dots\left(\boldsymbol{W}^{(1)} \otimes \boldsymbol{A}_{\text{sym}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}\right)\right\|_F}{\left\|\left(\boldsymbol{W}^{(l)} \otimes \boldsymbol{A}_{\text{sym}}\right)\dots\left(\boldsymbol{W}^{(1)} \otimes \boldsymbol{A}_{\text{sym}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,j}\right)\right\|_F} = \frac{\left\|\left(\boldsymbol{W}^{(l)}\dots\boldsymbol{W}^{(1)}\right) \otimes \left(\boldsymbol{A}_{\text{sym}}^l\boldsymbol{U}_{:,i}\right)\right\|_F}{\left\|\left(\boldsymbol{W}^{(l)}\dots\boldsymbol{W}^{(1)}\right) \otimes \left(\boldsymbol{A}_{\text{sym}}^l\boldsymbol{U}_{:,j}\right)\right\|_F}. \tag{3.63}$$

As in the proof of Theorem 3.2, the sub-multiplicativity of the norm of the Kronecker product and $\boldsymbol{U}_{:,i}$ being an eigenvector of $\boldsymbol{A}_{\text{sym}}$ allow the simplification

$$\frac{\left\|\left(\boldsymbol{W}^{(l)}\dots\boldsymbol{W}^{(1)}\right) \otimes \left(\boldsymbol{A}_{\text{sym}}^l\boldsymbol{U}_{:,i}\right)\right\|_F}{\left\|\left(\boldsymbol{W}^{(l)}\dots\boldsymbol{W}^{(1)}\right) \otimes \left(\boldsymbol{A}_{\text{sym}}^l\boldsymbol{U}_{:,j}\right)\right\|_F} = \frac{\left\|\boldsymbol{W}^{(l)}\dots\boldsymbol{W}^{(1)}\right\|_F \cdot \left\|\lambda_i^l\boldsymbol{U}_{:,i}\right\|_F}{\left\|\boldsymbol{W}^{(l)}\dots\boldsymbol{W}^{(1)}\right\|_F \cdot \left\|\lambda_j^l\boldsymbol{U}_{:,j}\right\|_F}. \tag{3.64}$$

The norm of the feature transformations cancels. By the absolute homogeneity property $\|\lambda_i \boldsymbol{U}_{:,i}\|_F = |\lambda_i| \cdot \|\boldsymbol{U}_{:,i}\|_F$ of the Frobenius norm, and $\|\boldsymbol{U}_{:,i}\| = \|\boldsymbol{U}_{:,j}\|$ due to orthonormality of $\boldsymbol{U}$. We have

$$\frac{\left\|\boldsymbol{W}^{(l)} \ldots \boldsymbol{W}^{(1)}\right\|_F \cdot \left\|(\lambda_i)^l \, \boldsymbol{U}_{:,i}\right\|_F}{\left\|\boldsymbol{W}^{(l)} \ldots \boldsymbol{W}^{(1)}\right\|_F \cdot \left\|(\lambda_j)^l \, \boldsymbol{U}_{:,j}\right\|_F} = \frac{|\,(\lambda_i)^l\,|}{|\,(\lambda_j)^l\,|}\,, \tag{3.65}$$

concluding the proof. □

This finding shows that repeatedly applying GCN iterations relatively amplifies the smooth component in the data by an exponential ratio in the number of iterations l. This holds for any feature transformations $\boldsymbol{W}^{(k)}$. As a consequence of this property, the subspace $\boldsymbol{S}_{(1)}$ corresponding to the largest absolute eigenvalue λ_1 and smooth eigenvector $\boldsymbol{U}_{:,1} = \boldsymbol{D}^{1/2}\mathbf{1}$ of $\boldsymbol{A}_{\text{sym}}$ dominates all other subspaces in the limit case:

Corollary 3.1 (Component Dominance of $\boldsymbol{S}_{(1)}$). Let $\boldsymbol{T}^{(k)} = \boldsymbol{W}^{(k)} \otimes \boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{nd \times nd}$, where $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d \times d}$ is any matrix and $\boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{n \times n}$ is the symmetrically normalized adjacency matrix with orthonormal eigenvectors $\boldsymbol{U} \in \mathbb{R}^{n \times n}$. Let $\boldsymbol{S}_{(i)} = \boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}$ with corresponding eigenvalue λ_i of $\boldsymbol{A}_{\text{sym}}$ for all $i \in [n]$. Then, for all $i \in [n]$,

$$\lim_{l \to \infty} \frac{\left\|\boldsymbol{T}^{(l)} \ldots \boldsymbol{T}^{(1)} \boldsymbol{S}_{(i)}\right\|_F}{\max_p \left\|\boldsymbol{T}^{(l)} \ldots \boldsymbol{T}^{(1)} \boldsymbol{S}_{(p)}\right\|_F} = \begin{cases} 0, & \text{if } i \neq 1 \\ 1, & \text{otherwise}\,, \end{cases} \tag{3.66}$$

and the rate of convergence is given by $\frac{\lambda_i}{\lambda_1}$ for $i \neq 1$.

Proof. Based on Proposition 3.4, we have

$$\lim_{l \to \infty} \frac{\left\|\boldsymbol{T}^{(l)} \ldots \boldsymbol{T}^{(1)} \boldsymbol{S}_{(i)}\right\|_F}{\max_p \left\|\boldsymbol{T}^{(l)} \ldots \boldsymbol{T}^{(1)} \boldsymbol{S}_{(p)}\right\|_F} = \lim_{l \to \infty} \frac{|\lambda_i^l|}{\max_p |\lambda_p^l|}\,. \tag{3.67}$$

As $\max_p \lambda_p = \lambda_1$ for $\boldsymbol{A}_{\text{sym}}$, we have $\max_p |\lambda_p^l| = |\lambda_1^l| = 1$. As $|\lambda_i| < |\lambda_1|$ for all $i \in [n]$, we have in the limit

$$\lim_{l \to \infty} \frac{|\lambda_i^l|}{\max_p |\lambda_p^l|} = \begin{cases} 1, & \text{if } i = 1 \\ 0, & \text{otherwise}\,. \end{cases} \tag{3.68}$$

The convergence rate is given by $|\lambda_i|/|\lambda_1|$ for $i \neq 1$. □

We can already explain the observed over-smoothing phenomenon with Proposition 3.4 and Corollary 3.1. In each GCN iteration, the basis vectors of each subspace corresponding to a graph Fourier basis vector are relatively scaled by their respective eigenvalue. This is irrespective of the employed feature transformation $\boldsymbol{W}^{(k)}$. When applying multiple GCN iterations, this scaling happens exponentially, and the graph Fourier components corresponding to λ_1 increasingly dominate all other graph Fourier components. We will define a more general form of this phenomenon in Definition 3.2 and refer to it as component dominance.

Up to this point, we only considered the basis vectors. Node representations are linear combinations of these basis vectors, i.e.,

$$\operatorname{vec}(\boldsymbol{X}^{(0)}) = \sum_{i=1}^{n} (\boldsymbol{I}_d \otimes \boldsymbol{U}_{:,i}) \operatorname{vec}(\boldsymbol{B}_{i,:}) \tag{3.69}$$

with $\boldsymbol{B} = \boldsymbol{U}^\top \boldsymbol{X}^{(0)}$. For each subspace $\mathcal{Q}_i$ corresponding to a graph Fourier basis vector $\boldsymbol{U}_{:,i}$, we have coefficients $\boldsymbol{B}_{i,:}$. As the differences in the norm of the basis vectors exponentially increase with the number of iterations l, the coefficients $\boldsymbol{B}_{i,:}$ need to be increasingly large compared to the coefficients $\boldsymbol{B}_{1,:}$ of the maximally amplified subspace $\mathcal{Q}_1$ to contribute non-negligibly to $\boldsymbol{X}^{(l)}$. As the feature transformation acts equivalently on each subspace $\mathcal{Q}_i$, the signals across subspaces need to be very different. The signal in $\mathcal{Q}_1$ needs to be exponentially small with increased depth in order not to dominate the representations. At the same time, the coefficients in other subspaces must be sufficiently large. This means that $\boldsymbol{X}^{(0)}$, or equivalently $\boldsymbol{B}_{1,:}$, needs to lie in an increasingly small subspace that converges to a measure-zero set under the Lebesgue measure for $l \to \infty$. The probability that $\boldsymbol{B}_{1,:}$ drawn from any bounded set lies inside this increasingly small subspace converges to zero. This corresponds to convergence in probability. Due to time-inhomogeneous feature transformations $\boldsymbol{W}^{(i)} \neq \boldsymbol{W}^{(j)}$ for every $i \neq j$, $\boldsymbol{X}^{(l)}$ may not converge for $l \to \infty$ but converges to the space $\mathcal{Q}_1$. This is known as convergence in probability (Dudley 2002), which we denote by plim. We formalize this in the following theorem:

Theorem 3.3 (Representations converge to $\mathcal{Q}_1$ in probability). Let $\boldsymbol{X}^{(k)} = \boldsymbol{A}_{\text{sym}} \boldsymbol{X}^{(k-1)} \boldsymbol{W}^{(k)}$, where $\boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{n \times n}$ is the symmetrically normalized adjacency matrix and $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d \times d}$ for every $k \in \mathbb{N}$. Then,

$$\operatorname*{plim}_{l \to \infty} \left\| \frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F} - \boldsymbol{M}^{(l)} \right\|_F = 0 \tag{3.70}$$

for $\boldsymbol{X}^{(0)} \in [-m, m]^{n \times d}$ for some $m \in \mathbb{R}$ with respect to the Lebesgue measure and some $\operatorname{vec}\left(\boldsymbol{M}^{(l)}\right) \in \mathcal{Q}_1$ that depends on l for every $l \in \mathbb{N}$.

Proof. By defining $\boldsymbol{O}^{(l)} = \boldsymbol{W}^{(1)} \dots \boldsymbol{W}^{(l)}$, we have

$$\boldsymbol{X}^{(l)} = \boldsymbol{A}_{\text{sym}}^l \boldsymbol{X}^{(0)} \boldsymbol{W}^{(1)} \dots \boldsymbol{W}^{(l)} = \boldsymbol{A}_{\text{sym}}^l \boldsymbol{X}^{(0)} \boldsymbol{O}^{(l)} \,. \tag{3.71}$$

Using properties of the Kronecker product, we can equivalently write this in vectorized form as $\left(\left(\boldsymbol{O}^{(l)}\right)^\top \otimes \boldsymbol{A}_{\text{sym}}^l\right) \operatorname{vec}(\boldsymbol{X}^{(0)})$. We omit the transpose for clarity. Using the singular value decomposition $\boldsymbol{O}^{(l)} = \boldsymbol{V}^{(l)} \boldsymbol{\Sigma}^{(l)} \left(\boldsymbol{N}^{(l)}\right)^\top$ and eigendecomposition $\boldsymbol{A}_{\text{sym}}^l = \boldsymbol{U} \boldsymbol{\Lambda}^l \boldsymbol{U}^\top$, we obtain

$$\operatorname{vec}\left(\boldsymbol{A}_{\text{sym}} \boldsymbol{X}^{(0)} \boldsymbol{O}^{(l)}\right) = \left(\boldsymbol{O}^{(l)} \otimes \boldsymbol{U}\right) \left(\boldsymbol{I}_d \otimes \boldsymbol{\Lambda}^l\right) \boldsymbol{b} \,, \tag{3.72}$$

where $\boldsymbol{b} = (\boldsymbol{I}_d \otimes \boldsymbol{U})^\top \operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)$. We now choose $\boldsymbol{m}^{(l)} = \frac{(\boldsymbol{O}^{(l)} \otimes \boldsymbol{U})(\boldsymbol{I}_d \otimes \boldsymbol{\Lambda}^l)\boldsymbol{b}_1}{\|\boldsymbol{X}^{(l)}\|_F}$ where $\boldsymbol{b}_1$ is equal to $\boldsymbol{b}$ for entries corresponding to $\mathcal{Q}_1$ and zero for all other entries. We thus have

$$\left\| \frac{\operatorname{vec}\left(\boldsymbol{X}^{(l)}\right)}{\left\|\operatorname{vec}\left(\boldsymbol{X}^{(l)}\right)\right\|_F} - \boldsymbol{m}^{(l)} \right\|_F = \left\| \frac{\left(\boldsymbol{O}^{(l)} \otimes \boldsymbol{U}\right)}{\left\|\boldsymbol{X}^{(l)}\right\|_F} \left(\boldsymbol{I}_d \otimes \boldsymbol{\Lambda}^l\right)(\boldsymbol{b} - \boldsymbol{b}_1) \right\|_F . \tag{3.73}$$

To bound the numerator and denominator, we divide both by the largest singular value $\sigma_1^{(l)}$ of $\boldsymbol{O}^{(l)}$, resulting in

$$\left\| \frac{\operatorname{vec}\left(\boldsymbol{X}^{(l)}\right)}{\left\|\operatorname{vec}\left(\boldsymbol{X}^{(l)}\right)\right\|_F} - \boldsymbol{m}^{(l)} \right\|_F = \left\| \frac{\left(\frac{\boldsymbol{O}^{(l)}}{\sigma_1^{(l)}} \otimes \boldsymbol{U}\right)}{\left\|\frac{\boldsymbol{X}^{(l)}}{\sigma_1^{(l)}}\right\|_F} \left(\boldsymbol{I}_d \otimes \boldsymbol{\Lambda}^l\right)(\boldsymbol{b} - \boldsymbol{b}_1) \right\|_F . \tag{3.74}$$

Using the sub-multiplicativity of the matrix norm and the absolute homogeneity, we upper this norm by the following product:

$$\left\| \frac{\operatorname{vec}\left(\boldsymbol{X}^{(l)}\right)}{\left\|\operatorname{vec}\left(\boldsymbol{X}^{(l)}\right)\right\|_F} - \boldsymbol{m}^{(l)} \right\|_F \leq \frac{\left\|\left(\frac{\boldsymbol{O}^{(l)}}{\sigma_1^{(l)}} \otimes \boldsymbol{U}\right)\right\|_2}{\left\|\frac{\boldsymbol{X}^{(l)}}{\sigma_1^{(l)}}\right\|_F} \left\|\left(\boldsymbol{I}_d \otimes \boldsymbol{\Lambda}^l\right)(\boldsymbol{b} - \boldsymbol{b}_1)\right\|_F \tag{3.75}$$

We now proceed by bounding each of the three norms separately. First, by the definition of the Frobenius norm, we have:

$$\left\|\left(\boldsymbol{I}_d \otimes \boldsymbol{\Lambda}^l\right)(\boldsymbol{b} - \boldsymbol{b}_1)\right\|_F = \sqrt{\sum_{i=1}^{n}\sum_{r=1}^{d} \left(\lambda_i^l \left(b_{i,r} - b'_{i,r}\right)\right)^2} . \tag{3.76}$$

As $b_{1,r} = b'_{1,r}$ for all $r \in [d]$ we can remove the terms where $i = 1$:

$$\left\|\left(\boldsymbol{I}_d \otimes \boldsymbol{\Lambda}^l\right)(\boldsymbol{b} - \boldsymbol{b}_1)\right\|_F = \sqrt{\sum_{i=2,r=1}^{n,d} \left(\lambda_i^l \left(b_{i,r} - b'_{i,r}\right)\right)^2} . \tag{3.77}$$

This can be further bounded by the largest absolute eigenvalue $\lambda_2^l < 1$:

$$\left\|\left(\boldsymbol{I}_d \otimes \boldsymbol{\Lambda}^l\right)(\boldsymbol{b} - \boldsymbol{b}_1)\right\|_F \leq \lambda_2^l \sqrt{\sum_{i=2,r=1}^{n,d} \left(b_{i,r} - b'_{i,r}\right)^2} . \tag{3.78}$$

The terms inside the square root can further be bounded by their maximal value $b_{\max} = \max_{i,r} |b_{i,r}|$:

$$\left\|\left(\boldsymbol{I}_d \otimes \boldsymbol{\Lambda}^l\right)(\boldsymbol{b} - \boldsymbol{b}_1)\right\|_F \leq \lambda_2^l \cdot b_{\max} \cdot \sqrt{(n-1) \cdot d} . \tag{3.79}$$

As $l \to \infty$, this part converges to zero as $\lim_{l\to\infty} \lambda_2^l = 0$ due to $|\lambda_2| < 1$.

Next, we bound the norm in the numerator:

$$\left\| \frac{\boldsymbol{O}^{(l)}}{\sigma_1^{(l)}} \otimes \boldsymbol{U} \right\|_2 = \left\| \frac{\boldsymbol{O}^{(l)}}{\sigma_1^{(l)}} \right\|_2 \cdot \|\boldsymbol{U}\|_2 = 1 \tag{3.80}$$

using the property that $\boldsymbol{U}$ and $\frac{\boldsymbol{O}^{(l)}}{\sigma_1^{(l)}}$ have a maximal singular value of 1.

As the last part, we bound the denominator:

$$\frac{1}{\left\| \frac{\mathrm{vec}(\boldsymbol{X}^{(l)})}{\sigma_1^{(l)}} \right\|_2} = \frac{1}{\sqrt{\frac{\mathrm{vec}(\boldsymbol{X}^{(l)})}{\sigma_1^{(l)}}^\top \frac{\mathrm{vec}(\boldsymbol{X}^{(l)})}{\sigma_1^{(l)}}}}, \tag{3.81}$$

which is given by the definition of the Frobenius norm. By substituting $\mathrm{vec}\left(\boldsymbol{X}^{(0)}\right) = \left(\boldsymbol{N}^{(l)} \otimes \boldsymbol{U}\right) \boldsymbol{b}^{(l)}$ where $\boldsymbol{b}^{(l)} = \left(\boldsymbol{N}^{(l)} \otimes \boldsymbol{U}\right)^\top \mathrm{vec}\left(\boldsymbol{X}^{(0)}\right)$, we get

$$\frac{1}{\left\| \frac{\mathrm{vec}(\boldsymbol{X}^{(l)})}{\sigma_1^{(l)}} \right\|_2} = \frac{1}{\sqrt{\left(\boldsymbol{b}^{(l)}\right)^\top \left(\frac{(\boldsymbol{N}^{(l)})^\top (\boldsymbol{O}^{(l)})^\top \boldsymbol{O}^{(l)} \boldsymbol{N}^{(l)}}{\left(\sigma_1^{(l)}\right)^2} \otimes \boldsymbol{U}^\top \boldsymbol{A}_{\mathrm{sym}}^{2l} \boldsymbol{U} \right) \boldsymbol{b}^{(l)}}}. \tag{3.82}$$

Due to orthonormality, we have $\boldsymbol{U}\boldsymbol{U}^\top = \boldsymbol{I}_n$ and $\boldsymbol{N}^{(l)} \left(\boldsymbol{N}^{(l)}\right)^\top = \boldsymbol{I}_d$. This allows the simplification

$$\frac{1}{\left\| \frac{\mathrm{vec}(\boldsymbol{X}^{(l)})}{\sigma_1^{(l)}} \right\|_2} = \frac{1}{\sqrt{\left(\boldsymbol{b}^{(l)}\right)^\top \left(\left(\frac{\boldsymbol{\Sigma}^{(l)}}{\left(\sigma_1^{(l)}\right)^2} \right)^2 \otimes \boldsymbol{\Lambda}^{2l} \right) \boldsymbol{b}^{(l)}}}. \tag{3.83}$$

As all involved matrices are diagonal, we can further simplify this as a sum of element-wise multiplications:

$$\frac{1}{\left\| \frac{\mathrm{vec}(\boldsymbol{X}^{(l)})}{\sigma_1^{(l)}} \right\|_2} = \frac{1}{\sqrt{\sum_{i=1}^n \sum_{r=1}^d \left(b_{i,r}^{(l)}\right)^2 \left(\frac{(\sigma_r^{(l)})^2}{\left(\sigma_1^{(l)}\right)^2} \right)^2 \lambda_i^{2l}}}. \tag{3.84}$$

This can be bounded by the coefficient $b_{1,1}$ corresponding to the maximal singular value $\sigma_1^{(l)}$ and the maximal eigenvalue λ_1 of $\boldsymbol{A}_{\mathrm{sym}}$:

$$\frac{1}{\left\| \frac{\mathrm{vec}(\boldsymbol{X}^{(l)})}{\sigma_1^{(l)}} \right\|_2} \leq \frac{1}{b_{1,1}^{(l)}} = \frac{1}{\left(\boldsymbol{N}_{:,1}^{(l)} \otimes \boldsymbol{U}_{:,1}\right)^\top \mathrm{vec}\left(\boldsymbol{X}^{(0)}\right)}. \tag{3.85}$$

Combining these upper bounds on each term, we get

$$\left\| \frac{\operatorname{vec}\left(\boldsymbol{X}^{(l)}\right)}{\left\|\operatorname{vec}\left(\boldsymbol{X}^{(l)}\right)\right\|_F} - \boldsymbol{m}^{(l)} \right\|_2 \leq \frac{1}{\left(\boldsymbol{N}_{:,1}^{(l)} \otimes \boldsymbol{U}_{:,1}\right)^\top \operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)} \lambda_2^l \cdot \sqrt{n-1} \cdot \sqrt{d} \cdot b_{\max} \,. \tag{3.86}$$

As the last step, we need to show that the probability of this term being larger than any fixed ϵ goes to zero for large l.

$$\begin{aligned} &\frac{1}{\left(\boldsymbol{N}_{:,1}^{(l)} \otimes \boldsymbol{U}_{:,1}\right)^\top \operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)} \cdot \lambda_2^l \cdot \sqrt{n-1} \cdot \sqrt{d} \cdot b_{\max} > \epsilon \\ \Leftrightarrow &\frac{1}{\epsilon} \cdot \lambda_2^l \cdot \sqrt{d} \cdot \sqrt{n-1} \cdot b_{\max} > \left(\boldsymbol{N}_{:,1}^{(l)} \otimes \boldsymbol{U}_{:,1}\right)^\top \operatorname{vec}\left(\boldsymbol{X}^{(0)}\right) \end{aligned} \tag{3.87}$$

As the left-hand side exponentially converges to zero, the right-hand side also converges to zero. Thus, the component $\left(\boldsymbol{N}_{:,1}^{(l)} \otimes \boldsymbol{U}_{:,1}\right)^\top \operatorname{vec}\left(\boldsymbol{X}^{(0)}\right) \in \mathbb{R}$ of the initial state needs to be increasingly close to zero. For a uniform distribution over any bounded interval, the probability of $\boldsymbol{X}^{(0)}$ being in this increasingly small subspace converges to zero. □

Theorem 3.3 confirms that the probability of having initial node representations $\boldsymbol{X}^{(0)}$ for which $\boldsymbol{X}^{(l)}$ is not getting dominated by the subspace $\mathcal{Q}_1$ converges to zero. As $\mathcal{Q}_1$ is constructed using the eigenvector $\boldsymbol{U}_{:,1} = \boldsymbol{D}^{1/2}\mathbf{1}$ of $\boldsymbol{A}_{\text{sym}}$, $\boldsymbol{X}^{(l)}$ gets dominated by this smooth vector in probability. Based on the extensive proof, this phenomenon can only be prevented when $\left(\boldsymbol{N}_{:,1}^{(l)} \otimes \boldsymbol{U}_{:,1}\right)^\top \operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)$, where $\boldsymbol{N}_{:,1}^{(l)}$ is the first right singular vector of $\boldsymbol{W}^{(1)} \dots \boldsymbol{W}^{(l)}$, is increasingly close to zero for $l \to \infty$.

Equivalently, this convergence can be stated in probability over the feature transformations $\boldsymbol{W}^{(k)}$ instead of the initial node representations $\boldsymbol{X}^{(0)}$. As the node representations need to align closely with the null space of the feature transformations, the probability can be taken over either one. Due to this property describing a relative phenomenon between components, the normalization of node representations emerges naturally. The shared scalar multiple $\left\|\boldsymbol{X}^{(l)}\right\|_F$ does not affect this phenomenon. When the feature transformations lead to a large norm $\left\|\boldsymbol{X}^{(l)}\right\|_F$, previous studies on over-smoothing claimed over-separation of node representations. Theorem 3.3 confirms that the smooth component still dominates the resulting node features. Studying the unnormalized node representations does not provide sufficient insights into this underlying phenomenon.

For the special case of a fixed and shared feature transformation $\boldsymbol{W} = \boldsymbol{W}^{(k)}$ for every $k \in \mathbb{N}$, $\boldsymbol{X}^{(0)}$ needs to align exactly with the null space of $\boldsymbol{W}$. This is a more specific requirement and corresponds to $\boldsymbol{X}^{(l)}$ being dominated by $\mathcal{Q}_1$ for almost every initial $\boldsymbol{X}^{(0)}$ with respect to the Lebesgue measure. We formalize this in the following Proposition:

Proposition 3.5 (Representations converge to $\mathcal{Q}_1$ for almost every $\boldsymbol{X}^{(0)}$). Let $\boldsymbol{X}^{(k)} = \boldsymbol{A}_{\text{sym}}\boldsymbol{X}^{(k-1)}\boldsymbol{W}$, where $\boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{n\times n}$ is the symmetrically normalized adjacency matrix and $\boldsymbol{W} \in \mathbb{R}^{d\times d}$ any matrix. Then,

$$\lim_{l\to\infty} \left\| \frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F} - \boldsymbol{M}^{(l)} \right\|_F = 0 \tag{3.88}$$

for almost every $\boldsymbol{X}^{(0)} \in \mathbb{R}^{n\times d}$ with respect to the Lebesgue measure and some $\text{vec}(\boldsymbol{M}^{(l)}) \in \mathcal{Q}_1$ that depends on l for every $l \in \mathbb{N}$.

Proof. Based on Theorem 3.3, we have

$$\left\| \frac{\text{vec}\left(\boldsymbol{X}^{(l)}\right)}{\left\|\text{vec}\left(\boldsymbol{X}^{(l)}\right)\right\|_F} - \boldsymbol{m}^{(l)} \right\|_F \leq \frac{1}{\left(\boldsymbol{N}_{:,1}\otimes \boldsymbol{U}_{:,1}\right)^\top \text{vec}\left(\boldsymbol{X}^{(0)}\right)} \cdot \lambda_2^l \cdot \sqrt{n-1}\cdot\sqrt{d}\cdot b_{\max}\,, \tag{3.89}$$

where $\boldsymbol{N}$ are the right singular vectors of $\boldsymbol{W}$ given by the singular value decomposition $\boldsymbol{W} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{N}^\top$. Thus, the right-hand side is a constant instead of a layer-dependent value. As such, this inequality holds for sufficiently large l when $\left(\boldsymbol{N}_{:,1}\otimes \boldsymbol{U}_{:,1}\right)^\top \text{vec}\left(\boldsymbol{X}^{(0)}\right) \neq 0$, which is satisfied for almost every $\boldsymbol{X}^{(0)}$. □

We summarize these two findings about $\boldsymbol{X}^{(l)}$ in the limit $l \to \infty$ as follows. For time-inhomogeneous feature transformation, the probability that the GCN avoids domination by the smooth subspace converges to zero as $l \to \infty$, but always remains non-zero for any finite l. When the feature transformation is shared across GCN iterations, this dominance occurs for almost every initial matrix $\boldsymbol{X}^{(0)} \in \mathbb{R}^{n\times d}$ with respect to the Lebesgue measure. Next, we show how this property can be equivalently observed as over-smoothing using the Dirichlet energy.

Over-Smoothing as a Consequence

We now connect our identified properties of the GCN with over-smoothing. Based on Theorem 3.3, we have identified that node representations $\boldsymbol{X}^{(l)}$ converge in probability to the space of rank-one matrices induced by $\mathcal{Q}_1$. Any $\text{vec}(\boldsymbol{M}) \in \mathcal{Q}_1$ is a rank-one matrix of the form $\boldsymbol{M} = \boldsymbol{U}_{:,1}\boldsymbol{b}_{(\boldsymbol{M})}$ with the dominant eigenvector of $\boldsymbol{A}_{\text{sym}}$ given by $\boldsymbol{U}_{:,1} = \boldsymbol{D}^{1/2}\boldsymbol{1}$ and column-wise scalars $\boldsymbol{b}_{(\boldsymbol{M})} \in \mathbb{R}^d$ that depend on $\boldsymbol{M}$. Thus, each column of $\boldsymbol{X}^{(l)}$ increasingly aligns with the degree-proportional vector. In spectral graph theory (see Section 2.2.2), $\boldsymbol{U}_{:,1}$ is the smoothest eigenvector of $\boldsymbol{L}_{\text{sym}}$, corresponding to the lowest frequency of the graph Fourier transform. As one common metric for observing over-smoothing (Cai et al. 2020; Zhou et al. 2021; Giovanni et al. 2023), the Dirichlet energy $E(\boldsymbol{X}) = \text{tr}\left(\boldsymbol{X}^\top \boldsymbol{L}_{\text{sym}}\boldsymbol{X}\right)$ has the property that the degree-proportional vector is in the null space of $\boldsymbol{L}_{\text{sym}}$, i.e., $\boldsymbol{L}_{\text{sym}}\boldsymbol{U}_{:,1} = \boldsymbol{0}$. Thus, we have $E(\boldsymbol{M}) = 0$ for any $\text{vec}(\boldsymbol{M}) \in \mathcal{Q}_1$. As $\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}$ converges in probability to matrices $\text{vec}(\boldsymbol{M}) \in \mathcal{Q}_1$ as $l \to \infty$, we can equivalently show and observe that the Dirichlet energy $E\left(\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}\right)$ of the normalized node representations converges to zero in probability. We formalize this in the following proposition:

Proposition 3.6 (The linearized GCN exhibits Over-Smoothing). Let $\boldsymbol{X}^{(k)} = \boldsymbol{A}_{\text{sym}}\boldsymbol{X}^{(k-1)}\boldsymbol{W}^{(k)}$, where $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d\times d}$ is any matrix for all $k \in \mathbb{N}$ and $\boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{n\times n}$ is the symmetrically normalized adjacency matrix with λ_2 being its second largest eigenvalue in magnitude. Then,

$$\operatorname*{plim}_{l\to\infty} E\left(\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}\right) = 0 \tag{3.90}$$

with convergence rate λ_2^2 in probability for $\boldsymbol{X}^{(0)} \in [-m, m]^{n\times d}$ for any $m \in \mathbb{R}$ with respect to the Lebesgue measure.

Proof. Recall the definition of the Dirichlet energy and the Frobenius norm:

$$E\left(\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}\right) = \frac{\operatorname{tr}\left(\left(\boldsymbol{X}^{(l)}\right)^\top \boldsymbol{L}_{\text{sym}}\boldsymbol{X}^{(l)}\right)}{\operatorname{tr}\left(\left(\boldsymbol{X}^{(l)}\right)^\top \boldsymbol{X}^{(l)}\right)}. \tag{3.91}$$

Using the identity $\operatorname{tr}\left(\boldsymbol{A}^\top\boldsymbol{B}\right) = \operatorname{vec}(\boldsymbol{A})^\top \operatorname{vec}(\boldsymbol{B})$ for any matrices $\boldsymbol{A}, \boldsymbol{B}$ of suitable shape yields

$$E\left(\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}\right) = \frac{\operatorname{vec}\left(\boldsymbol{X}^{(l)}\right)^\top \operatorname{vec}\left(\boldsymbol{L}_{\text{sym}}\boldsymbol{X}^{(l)}\right)}{\operatorname{vec}\left(\boldsymbol{X}^{(l)}\right)^\top \operatorname{vec}\left(\boldsymbol{X}^{(l)}\right)}. \tag{3.92}$$

Let $\boldsymbol{O}^{(l)} = \boldsymbol{W}^{(1)}\ldots\boldsymbol{W}^{(l)}$. Based on the identity $\operatorname{vec}\left(\boldsymbol{X}^{(l)}\right) = \left(\boldsymbol{O}^{(l)} \otimes \boldsymbol{A}_{\text{sym}}^l\right)\operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)$, we obtain

$$E\left(\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}\right) = \frac{\operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)^\top \left(\boldsymbol{O}^{(l)} \otimes \boldsymbol{A}_{\text{sym}}^l\right)^\top \left(\boldsymbol{I} \otimes \boldsymbol{L}_{\text{sym}}\right)\left(\boldsymbol{O}^{(l)} \otimes \boldsymbol{A}_{\text{sym}}^l\right)\operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)}{\operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)^\top \left(\boldsymbol{O}^{(l)} \otimes \boldsymbol{A}_{\text{sym}}^l\right)^\top \left(\boldsymbol{O}^{(l)} \otimes \boldsymbol{A}_{\text{sym}}^l\right)\operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)}. \tag{3.93}$$

Consider the singular value decomposition $\boldsymbol{O}^{(l)} = \boldsymbol{V}^{(l)}\boldsymbol{\Sigma}^{(l)}\left(\boldsymbol{N}^{(l)}\right)^\top$ and eigendecomposition $\boldsymbol{A}_{\text{sym}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$. Using the singular vectors and eigenvectors as basis vectors, we rewrite the vectorized node representations as $\operatorname{vec}\left(\boldsymbol{X}^{(0)}\right) = \left(\boldsymbol{N}^{(l)} \otimes \boldsymbol{U}\right)\operatorname{vec}\left(\boldsymbol{B}^{(l)}\right)$ for $\operatorname{vec}\left(\boldsymbol{B}^{(l)}\right) = \left(\boldsymbol{N}^{(l)} \otimes \boldsymbol{U}\right)^\top \operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)$. We also utilize the mixed-product property $(\boldsymbol{A}\otimes\boldsymbol{B})(\boldsymbol{C}\otimes\boldsymbol{D}) = (\boldsymbol{A}\boldsymbol{C}\otimes\boldsymbol{B}\boldsymbol{D})$ for any matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ of suitable shape. This yields

$$E\left(\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}\right) = \frac{\left(\operatorname{vec}\left(\boldsymbol{B}^{(l)}\right)\right)^\top \left(\left(\boldsymbol{N}^{(l)}\right)^\top \left(\boldsymbol{O}^{(l)}\right)^\top \boldsymbol{O}^{(l)}\boldsymbol{N}^{(l)}\right) \otimes \left(\left(\boldsymbol{U}\right)^\top \boldsymbol{A}_{\text{sym}}^l \boldsymbol{L}_{\text{sym}} \boldsymbol{A}_{\text{sym}}^l \boldsymbol{U}\right)\operatorname{vec}\left(\boldsymbol{B}^{(l)}\right)}{\left(\operatorname{vec}\left(\boldsymbol{B}^{(l)}\right)\right)^\top \left(\left(\boldsymbol{N}^{(l)}\right)^\top \left(\boldsymbol{O}^{(l)}\right)^\top \boldsymbol{O}^{(l)}\boldsymbol{N}^{(l)}\right) \otimes \left(\left(\boldsymbol{U}\right)^\top \boldsymbol{A}_{\text{sym}}^l \boldsymbol{A}_{\text{sym}}^l \boldsymbol{U}\right)\operatorname{vec}\left(\boldsymbol{B}^{(l)}\right)}. \tag{3.94}$$

As $\boldsymbol{O}^{(l)}\boldsymbol{N}^{(l)} = \boldsymbol{V}^{(l)}\boldsymbol{\Sigma}^{(l)}$ and $\boldsymbol{A}_{\text{sym}}^{l}\boldsymbol{U} = \boldsymbol{U}\boldsymbol{\Lambda}^{l}$, this simplifies as

$$E\left(\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}\right) = \frac{\operatorname{vec}\left(\boldsymbol{B}^{(l)}\right)^{\top}\left(\left(\boldsymbol{\Sigma}^{(l)}\right)^{2} \otimes \boldsymbol{\Lambda}^{2l}\left(\boldsymbol{I}_n - \boldsymbol{\Lambda}\right)\right)\operatorname{vec}\left(\boldsymbol{B}^{(l)}\right)}{\operatorname{vec}\left(\boldsymbol{B}^{(l)}\right)^{\top}\left(\left(\boldsymbol{\Sigma}^{(l)}\right)^{2} \otimes \boldsymbol{\Lambda}^{2l}\right)\operatorname{vec}\left(\boldsymbol{B}^{(l)}\right)}\,. \tag{3.95}$$

As all involved matrices are diagonal, this can be simplified as a sum over element-wise products:

$$E\left(\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}\right) = \frac{\sum_{i=1}^{n}\sum_{r=1}^{d}\left(B_{i,r}^{(l)}\right)^{2}\left(\sigma_r^{(l)}\right)^{2}\lambda_i^{2}\left(1-\lambda_i\right)}{\sum_{i=1}^{n}\sum_{r=1}^{d}\left(B_{i,r}^{(l)}\right)^{2}\left(\sigma_r^{(l)}\right)^{2}\lambda_i^{2}}\,, \tag{3.96}$$

where $\sigma_r^{(l)} = \Sigma_{r,r}^{(l)}$ and $\lambda_i = \Lambda_{i,i}$. We normalize the singular values using the largest singular value $\sigma_1^{(l)}$ in both the numerator and denominator. As the maximal eigenvalue $\lambda_1 = 1$ is one, normalization of eigenvalues is not required:

$$E\left(\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}\right) = \frac{\sum_{i=1}^{n}\sum_{r=1}^{d}\left(B_{i,r}^{(l)}\right)^{2}\frac{\left(\sigma_r^{(l)}\right)^{2}}{\left(\sigma_1^{(l)}\right)^{2}}\lambda_i^{2}\left(1-\lambda_i\right)}{\sum_{i=1}^{n}\sum_{r=1}^{d}\left(B_{i,r}^{(l)}\right)^{2}\frac{\left(\sigma_r^{(l)}\right)^{2}}{\left(\sigma_1^{(l)}\right)^{2}}\lambda_i^{2}}\,. \tag{3.97}$$

We bound both the numerator using the maximal absolute coefficient $b_{\max}^{(l)} = \max_{i,r}|B_{i,r}^{(l)}|$ and the denominator using coefficient $B_{1,1}^{(l)}$ corresponding to the first eigenvalue $\lambda_1 = 1$ and singular value $\sigma_1^{(l)}/\sigma_1^{(l)} = 1$:

$$E\left(\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}\right) < \frac{(n-1)\cdot d\cdot\left(b_{\max}^{(l)}\right)^{2}\lambda_2^{2l}\left(1-\lambda_2\right)}{\left(B_{1,1}^{(l)}\right)^{2}}\,. \tag{3.98}$$

For convergence in probability, we need to ensure that the probability that this upper bound is larger than any fixed epsilon converges to zero, i.e.,

$$\lim_{l\to\infty}\mathbb{P}\left(\frac{(n-1)\cdot d\cdot(b_{\max}^{(l)})^{2}\lambda_2^{2l}\left(1-\lambda_2\right)}{\left(B_{1,1}^{(l)}\right)^{2}} > \epsilon\right) = 0\,. \tag{3.99}$$

As $\lambda_2^{2l} = 0$ for $l \to \infty$, it suffices to show that the ratio $\frac{\left(b_{\max}^{(l)}\right)^{2}}{\left(B_{1,1}^{(l)}\right)^{2}}$ is bounded in probability. Given that $\boldsymbol{X}^{(0)}$ is sampled from a bounded domain, the probability that a randomly sampled $\boldsymbol{X}^{(0)}$ violates this condition converges to zero. □

Based on this finding, the probability that over-smoothing, as measured by the Dirichlet energy, occurs for $\boldsymbol{X}^{(l)}$ converges to one for $l \to \infty$. This finding further motivates applying the Dirichlet energy to normalized node representations $\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}$ to empirically observe over-smoothing and its underlying theoretical property. Contrarily, applying the Dirichlet energy to the unnormalized node representations $\boldsymbol{X}^{(l)}$ does not capture this property. For unnormalized node representations, we could not distinguish between avoiding over-smoothing and the smooth component dominating $\boldsymbol{X}^{(l)}$ while all other components are negligible in comparison. This also aligns with our findings from Section 3.3.3, in which we have shown that the norm of the node representations overshadows possible insights of the Dirichlet energy when applied to unnormalized node representations. It also aligns with our spectral study on the GCN in Section 3.2.2, in which we have shown that all filters of the GCN in the SISO case are equivalent up to a uniform scaling of all components. Previous work similarly uses the normalized state within the Dirichlet energy for improved insights into the behavior of MPNNs (D. Chen et al. 2020; Liu et al. 2020; Giovanni et al. 2023; Maskey et al. 2023). Next, we generalize our study to arbitrary aggregation matrices $\tilde{\boldsymbol{A}}$ instead of the symmetrically normalized adjacency matrix $\boldsymbol{A}_{\text{sym}}$.

3.5.2 The Effect of Arbitrary Aggregations

To this point, our theoretical study assumed the aggregation matrix to be the symmetrically normalized adjacency matrix $\boldsymbol{A}_{\text{sym}}$. However, many other MPNNs adopt a similar update rule to Equation (3.42) that can be expressed

$$\boldsymbol{X}^{(k)} = \tilde{\boldsymbol{A}}\boldsymbol{X}^{(k-1)}\boldsymbol{W}^{(k)}\,, \tag{3.100}$$

where $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n \times n}$ is an arbitrary matrix containing any edge weights between any two nodes. Examples of such an $\tilde{\boldsymbol{A}}$ include the sum aggregation, mean aggregation, attention-based scores, or negative edge weights. As described in Section 3.2, most previous studies assumed a fixed aggregation function, typically $\boldsymbol{A}_{\text{sym}}$ or $\boldsymbol{A}_{\text{rw}}$. We find that the insights from Theorem 3.2 (shared component amplification), Corollary 3.1 (component dominance), and Theorem 3.3 extend to this generalized case.

A Single Iteration

Specifically, we first extend Theorem 3.2 to arbitrary aggregation functions $\tilde{\boldsymbol{A}}$ and prove that feature transformations cannot influence the relative amplification of components whenever an update rule as in Equation (3.100) is used:

Proposition 3.7 (Component Amplification is independent of $\boldsymbol{W}$). Let $\boldsymbol{T} = \boldsymbol{W} \otimes \tilde{\boldsymbol{A}} \in \mathbb{R}^{nd \times nd}$, where $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ and $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n \times n}$ are arbitrary matrices. Let $\boldsymbol{P}_{(i)} \in \mathbb{R}^{n \times n_{(i)}}$ and $\boldsymbol{P}_{(j)} \in \mathbb{R}^{n \times n_{(j)}}$ be two arbitrary matrices for some $n_{(i)}, n_{(j)} \in \mathbb{N}$. Then,

$$\frac{\left\|\left(\boldsymbol{W} \otimes \tilde{\boldsymbol{A}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{P}_{(i)}\right)\right\|_F}{\left\|\left(\boldsymbol{W} \otimes \tilde{\boldsymbol{A}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{P}_{(j)}\right)\right\|_F} = \frac{\left\|\tilde{\boldsymbol{A}}\boldsymbol{P}_{(i)}\right\|_F}{\left\|\tilde{\boldsymbol{A}}\boldsymbol{P}_{(j)}\right\|_F}. \tag{3.101}$$

Proof. The proof is analogous to the proof of Theorem 3.2. We begin by applying the mixed-product property of the Kronecker and matrix product:

$$\frac{\left\|\left(\boldsymbol{W} \otimes \tilde{\boldsymbol{A}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{P}_{(i)}\right)\right\|_F}{\left\|\left(\boldsymbol{W} \otimes \tilde{\boldsymbol{A}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{P}_{(j)}\right)\right\|_F} = \frac{\left\|\boldsymbol{W} \otimes \tilde{\boldsymbol{A}}\boldsymbol{P}_{(i)}\right\|_F}{\left\|\boldsymbol{W} \otimes \tilde{\boldsymbol{A}}\boldsymbol{P}_{(j)}\right\|_F}. \tag{3.102}$$

Now, we use the sub-multiplicativity of the norm for Kronecker products:

$$\frac{\left\|\boldsymbol{W} \otimes \tilde{\boldsymbol{A}}\boldsymbol{P}_{(i)}\right\|_F}{\left\|\boldsymbol{W} \otimes \tilde{\boldsymbol{A}}\boldsymbol{P}_{(j)}\right\|_F} = \frac{\|\boldsymbol{W}\|_F \cdot \left\|\tilde{\boldsymbol{A}}\boldsymbol{P}_{(i)}\right\|_F}{\|\boldsymbol{W}\|_F \cdot \left\|\tilde{\boldsymbol{A}}\boldsymbol{P}_{(j)}\right\|_F}. \tag{3.103}$$

Since $\|\boldsymbol{W}\|_F$ cancels out, we conclude:

$$\frac{\left\|\left(\boldsymbol{W} \otimes \tilde{\boldsymbol{A}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{P}_{(i)}\right)\right\|_F}{\left\|\left(\boldsymbol{W} \otimes \tilde{\boldsymbol{A}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{P}_{(j)}\right)\right\|_F} = \frac{\left\|\tilde{\boldsymbol{A}}\boldsymbol{P}_{(i)}\right\|_F}{\left\|\tilde{\boldsymbol{A}}\boldsymbol{P}_{(j)}\right\|_F} \tag{3.104}$$

□

Matrices $\boldsymbol{P}_{(i)}$ and $\boldsymbol{P}_{(j)}$ correspond to bases of different components in the node representations. The relative amplification of these components solely depends on the aggregation matrix $\tilde{\boldsymbol{A}}$. It is thus fixed and independent of the learnable feature transformation $\boldsymbol{W}$. As with Theorem 3.2, it is also shared across feature channels, meaning that the same component gets amplified for all feature channels. We will define a generalized form of this phenomenon of shared component amplification (SCA) in Definition 3.1. For full-rank matrices $\tilde{\boldsymbol{A}}$, $\boldsymbol{P}_{(i)}$ can be constructed using their eigenvectors. In the general case for any $\tilde{\boldsymbol{A}}$, $\boldsymbol{P}_{(i)} = \boldsymbol{V}_{:,i}$ can be selected as the singular vector $\boldsymbol{V}_{:,i}$ utilizing the singular value decomposition $\tilde{\boldsymbol{A}} = \boldsymbol{N}\boldsymbol{\Sigma}\boldsymbol{V}^\top$. Each component then gets amplified by its corresponding singular value. We formalize this property in the following corollary:

Theorem 3.4 (SCA in Message-Passing). Let $\boldsymbol{T} = \boldsymbol{W} \otimes \tilde{\boldsymbol{A}} \in \mathbb{R}^{nd \times nd}$, where $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ and $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n \times n}$ are arbitrary matrices. Let the singular value decomposition of $\tilde{\boldsymbol{A}}$ be $\tilde{\boldsymbol{A}} = \boldsymbol{N}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ with $\boldsymbol{V}_{:,i} \in \mathbb{R}^{n \times 1}$ and $\sigma_i = \Sigma_{i,i}$ for all $i \in [n]$. Then, for all $i, j \in [n]$,

$$\frac{\left\|\left(\boldsymbol{W} \otimes \tilde{\boldsymbol{A}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{V}_{:,i}\right)\right\|_F}{\left\|\left(\boldsymbol{W} \otimes \tilde{\boldsymbol{A}}\right)\left(\boldsymbol{I}_d \otimes \boldsymbol{V}_{:,j}\right)\right\|_F} = \frac{\sigma_i}{\sigma_j}. \tag{3.105}$$

Proof. Following Proposition 3.7, we have

$$\frac{\left\|\left(\boldsymbol{W}\otimes\tilde{\boldsymbol{A}}\right)(\boldsymbol{I}_d\otimes\boldsymbol{V}_{:,i})\right\|_F}{\left\|\left(\boldsymbol{W}\otimes\tilde{\boldsymbol{A}}\right)(\boldsymbol{I}_d\otimes\boldsymbol{V}_{:,j})\right\|_F}=\frac{\left\|\tilde{\boldsymbol{A}}\boldsymbol{V}_{:,i}\right\|_F}{\left\|\tilde{\boldsymbol{A}}\boldsymbol{V}_{:,j}\right\|_F} \tag{3.106}$$

As each $\boldsymbol{V}_{:,i}$ is a right singular vector of $\tilde{\boldsymbol{A}}$, we have

$$\frac{\left\|\tilde{\boldsymbol{A}}\boldsymbol{V}_{:,i}\right\|_F}{\left\|\tilde{\boldsymbol{A}}\boldsymbol{V}_{:,j}\right\|_F}=\frac{\left\|\sigma_i\boldsymbol{N}_{:,i}\right\|_F}{\left\|\sigma_j\boldsymbol{N}_{:,j}\right\|_F}. \tag{3.107}$$

As $\boldsymbol{N}$ is an orthonormal matrix, this reduces to

$$\frac{\left\|\sigma_i\boldsymbol{N}_{:,i}\right\|_F}{\left\|\sigma_j\boldsymbol{N}_{:,j}\right\|_F}=\frac{\sigma_i}{\sigma_j}. \tag{3.108}$$

□

Based on the singular vectors $\boldsymbol{V}$ of the aggregation matrix $\tilde{\boldsymbol{A}}$, the singular vector $\boldsymbol{V}_{:,1}$ with the maximal singular value σ_1 gets amplified maximally with a fixed ratio relative to all other singular vectors across all feature channels. As the subspaces induced by the singular vectors are not invariant to $\boldsymbol{W}\otimes\tilde{\boldsymbol{A}}$, Theorem 3.4 cannot be extended to the iterated case. However, for any matrix, $\boldsymbol{P}_{(i)}$ can be selected to be generalized eigenvectors of $\tilde{\boldsymbol{A}}$ based on the Jordan normal form. For all eigenvalues of the same magnitude, the corresponding generalized eigenvectors span subspaces that are invariant under $\tilde{\boldsymbol{A}}$.

Repeated Iterations

We consider the Jordan decomposition $\tilde{\boldsymbol{A}}=\boldsymbol{P}\boldsymbol{J}\boldsymbol{P}^{-1}$ where $\boldsymbol{J}\in\mathbb{R}^{n\times n}$ is a block diagonal matrix that contains the Jordan blocks of $\tilde{\boldsymbol{A}}$ and $\boldsymbol{P}$ contains the generalized eigenvectors. Let $\boldsymbol{J}_{(i)}$ contain the Jordan blocks corresponding to all eigenvalues with the same magnitude. We form the basis vectors $\boldsymbol{P}_{(i)}$ that contain all columns of the generalized eigenvectors $\boldsymbol{P}$ corresponding to all eigenvalues of $\tilde{\boldsymbol{A}}$ with the same magnitude. Each $\boldsymbol{P}_{(i)}$ spans a subspace that is invariant under $\tilde{\boldsymbol{A}}$, i.e., we have $\tilde{\boldsymbol{A}}\boldsymbol{P}_{(i)}=\boldsymbol{P}_{(i)}\boldsymbol{J}_{(i)}$. It is also known that the Jordan block $\boldsymbol{J}_{(i)}$ containing the eigenvalues with the largest magnitude dominates all other Jordan blocks $\boldsymbol{J}_{(j)}$ in the limit $l\to\infty$. We formalize this in the following proposition:

Proposition 3.8 (Component Dominance for Arbitrary Aggregation Matrices). Let $\boldsymbol{T}^{(k)}=\boldsymbol{W}^{(k)}\otimes\tilde{\boldsymbol{A}}\in\mathbb{R}^{nd\times nd}$ for $k\in\mathbb{N}$, where $\boldsymbol{W}^{(k)}\in\mathbb{R}^{d\times d}$ are arbitrary matrices and $\tilde{\boldsymbol{A}}\in\mathbb{R}^{n\times n}$ has Jordan decomposition $\tilde{\boldsymbol{A}}=\boldsymbol{P}\boldsymbol{J}\boldsymbol{P}^{-1}$. Let r denote the number of distinct eigenvalue magnitudes of $\tilde{\boldsymbol{A}}$, and for each $i\in[r]$, let $\boldsymbol{P}_{(i)}\in\mathbb{R}^{n\times d_{(i)}}$ be the submatrix of

$\boldsymbol{P}$ consisting of the $d_{(i)}$ columns corresponding to eigenvalues of $\tilde{\boldsymbol{A}}$ with magnitude $|\lambda_i|$. Then, for all $i \in [n]$,

$$\frac{\left\|\left(\boldsymbol{W}^{(l)} \otimes \tilde{\boldsymbol{A}}\right) \dots \left(\boldsymbol{W}^{(1)} \otimes \tilde{\boldsymbol{A}}\right) \left(\boldsymbol{I}_d \otimes \boldsymbol{P}_{(i)}\right)\right\|_F}{\max_p \left\|\left(\boldsymbol{W}^{(l)} \otimes \tilde{\boldsymbol{A}}\right) \dots \left(\boldsymbol{W}^{(1)} \otimes \tilde{\boldsymbol{A}}\right) \left(\boldsymbol{I}_d \otimes \boldsymbol{P}_{(p)}\right)\right\|_F} = \begin{cases} 1, & \text{if } |\lambda_i| = \max_{j\in[n]} |\lambda_j| \\ 0, & \text{otherwise}\,. \end{cases} \tag{3.109}$$

Proof. Without loss of generality, we assume $\max_{j\in[n]} |\lambda_j| = |\lambda_1|$. We have $|\lambda_i| < |\lambda_1|$ for all $i \neq 1$. Based on Proposition 3.7, we obtain

$$\frac{\left\|\left(\boldsymbol{W}^{(l)} \otimes \tilde{\boldsymbol{A}}\right) \dots \left(\boldsymbol{W}^{(1)} \otimes \tilde{\boldsymbol{A}}\right) \left(\boldsymbol{I}_d \otimes \boldsymbol{P}_{(i)}\right)\right\|_F}{\max_p \left\|\left(\boldsymbol{W}^{(l)} \otimes \tilde{\boldsymbol{A}}\right) \dots \left(\boldsymbol{W}^{(1)} \otimes \tilde{\boldsymbol{A}}\right) \left(\boldsymbol{I}_d \otimes \boldsymbol{P}_{(p)}\right)\right\|_F} = \frac{\left\|\boldsymbol{P}_{(i)} \boldsymbol{J}_{(i)}^l\right\|_F}{\max_p \left\|\boldsymbol{P}_{(p)} \boldsymbol{J}_{(p)}^l\right\|_F}\,. \tag{3.110}$$

We bound the numerator and denominator using the largest eigenvalue in magnitude $|\lambda_1|$ of $\tilde{\boldsymbol{A}}$ to obtain

$$\frac{\left\|\boldsymbol{P}_{(i)} \boldsymbol{J}_{(i)}^l\right\|_F}{\max_p \left\|\boldsymbol{P}_{(p)} \boldsymbol{J}_{(p)}^l\right\|_F} = \frac{\left\|\boldsymbol{P}_{(i)} \left(\frac{\boldsymbol{J}_{(i)}}{|\lambda_1|}\right)^l\right\|_F}{\max_p \left\|\boldsymbol{P}_{(p)} \left(\frac{\boldsymbol{J}_{(p)}}{|\lambda_1|}\right)^l\right\|_F}\,. \tag{3.111}$$

We now only consider the limit behavior of the numerator. For $i \neq 1$ all diagonal entries of $\boldsymbol{J}_{(i)}$ are less than one in magnitude, all entries of $\left(\frac{\boldsymbol{J}_{(i)}}{|\lambda_1|}\right)^l$ converge to zero. Thus, we have for $i \neq 1$,

$$\lim_{l\to\infty} \left\|\boldsymbol{P}_{(i)} \left(\frac{\boldsymbol{J}_{(i)}}{|\lambda_1|}\right)^l\right\|_F = 0\,. \tag{3.112}$$

We now consider the case $i = 1$. Let $\boldsymbol{v}^\top = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}$. We have $\boldsymbol{J}_{(i)}\boldsymbol{v} = \boldsymbol{v}$ and $\boldsymbol{P}_{(i)} v = (\boldsymbol{P}_{(i)})_{:,1}$. By definition, we can set $\left\|\left(\boldsymbol{P}_{(i)}\right)_{:,1}\right\|_2 = 1$. We also use the norm inequality $\|\cdot\|_F \geq \|\cdot\|_2$. In total, we have

$$\left\|\boldsymbol{P}_{(1)} \left(\frac{\boldsymbol{J}_{(1)}}{|\lambda_1|}\right)^l\right\|_2 = \max_{\|u\|_2 \leq 1} \left\|\boldsymbol{P}_{(1)} \left(\frac{\boldsymbol{J}_{(1)}}{|\lambda_1|}\right)^l \boldsymbol{u}\right\|_2 \geq \left\|\boldsymbol{P}_{(1)} \left(\frac{\boldsymbol{J}_{(1)}}{|\lambda_1|}\right)^l \boldsymbol{v}\right\|_2 = \left\|\left(\boldsymbol{P}_{(1)}\right)_{:,1}\right\|_2 \tag{3.113}$$

Thus, $i = 1$ is the only index that does not converge to zero, which is therefore maximal. The numerator converges to zero for all other cases, while the denominator is lower bounded by 1. This concludes the proof. □

This confirms that the components of $\boldsymbol{X}^{(0)}$ corresponding to the largest eigenvalue in magnitude get dominantly amplified compared to the components corresponding to all other eigenvalues. Unless the initial components corresponding to $\boldsymbol{P}_{(1)}$ have increasingly small magnitude, node representations $\boldsymbol{X}^{(l)}$ will be dominated by the components from $\boldsymbol{P}_{(1)}$. This is equivalent to Corollary 3.1. We can similarly show convergence in probability to subspace $\boldsymbol{P}_{(1)}$.

Theorem 3.5 (Representations converge to $\mathcal{Q}_1$ in probability). Let $\boldsymbol{X}^{(k)} = \tilde{\boldsymbol{A}}\boldsymbol{X}^{(k-1)}\boldsymbol{W}^{(k)}$, where $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n\times n}$ is a matrix with Jordan decomposition $\tilde{\boldsymbol{A}} = \boldsymbol{P}\boldsymbol{J}\boldsymbol{P}^{-1}$, and each $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d\times d}$ for every $k \in \mathbb{N}$. Let $\boldsymbol{P}_{(1)} \in \mathbb{R}^{n\times d_{(1)}}$ denote the submatrix of $\boldsymbol{P}$ consisting of the $d_{(1)}$ columns corresponding to eigenvalues of $\tilde{\boldsymbol{A}}$ whose magnitude is equal to $\max_{i\in[n]} |\lambda_i|$. Then,

$$\operatorname*{plim}_{l\to\infty} \left\| \frac{\boldsymbol{X}^{(l)}}{\|\boldsymbol{X}^{(l)}\|_F} - \boldsymbol{P}_{(1)}\boldsymbol{V}^{(l)} \right\|_F = 0 \tag{3.114}$$

for $\boldsymbol{X}^{(0)} \in [-m, m]^{n\times d}$ sampled with respect to the Lebesgue measure for some $m \in \mathbb{R}$ and some $\boldsymbol{V}^{(l)} \in \mathbb{R}^{d_{(1)}\times d}$ for each $l \in \mathbb{N}$.

Proof. Based on the proof of Theorem 3.3, we can upper bound the distance to a rank-one matrix using the inequality

$$\left\| \frac{\boldsymbol{X}^{(l)}}{\|\boldsymbol{X}^{(l)}\|_F} - \boldsymbol{P}_{(1)}\boldsymbol{V}^{(l)} \right\|_F \le \frac{\left\| \frac{\boldsymbol{O}^{(l)}}{\sigma_1^{(l)}} \otimes \boldsymbol{P} \right\|_2}{\left\| \frac{\boldsymbol{X}^{(l)}}{\lambda_1 \sigma_1^{(l)}} \right\|_F} \left\| \left(\boldsymbol{I}_d \otimes \frac{\boldsymbol{J}^l}{|\lambda_1|} \right) (\boldsymbol{b} - \boldsymbol{b}_1) \right\|_2 \tag{3.115}$$

where $\sigma_1^{(l)}$ is the largest singular value of $\boldsymbol{O}^{(l)} = \boldsymbol{W}^{(l)} \dots \boldsymbol{W}^{(1)}$, $\boldsymbol{b} = (\boldsymbol{I}_d \otimes \boldsymbol{U})^\top \operatorname{vec}\left(\boldsymbol{X}^{(0)}\right) \in \mathbb{R}^{nd}$, and $\boldsymbol{b}_1 \in \mathbb{R}^{nd}$ is a copy of $\boldsymbol{b}$ that contains zeros for all positions $(b_1)_i$ for which $|J_{i,i}| \neq |\lambda_1|$. Equivalently to the proof of Theorem 3.3, we provide upper bounds for each of the three norms separately. First, we bound the Jordan block matrix $\boldsymbol{J}$ using the second-largest eigenvalue in magnitude $|\lambda_2|$:

$$\left\| \left(\boldsymbol{I}_d \otimes \frac{\boldsymbol{J}^l}{|\lambda_1|} \right) (\boldsymbol{b} - \boldsymbol{b}_1) \right\|_2 \le n \cdot l^{n-1} \cdot \left(\frac{|\lambda_2|}{|\lambda_1|} \right)^l \cdot b_{\max} \cdot \sqrt{n \cdot d}\,. \tag{3.116}$$

Next, we bound the nominator of the fraction using the largest entry of $\boldsymbol{P}$:

$$\left\| \frac{\boldsymbol{O}^{(l)}}{\sigma_1^{(l)}} \otimes \boldsymbol{P} \right\|_2 = n \cdot \max_{i,j} P_{i,j}\,. \tag{3.117}$$

Equivalently to the proof of Theorem 3.3, we bound the denominator using a component of $\boldsymbol{X}^{(l)}$ corresponding to an eigenvalue of $\tilde{\boldsymbol{A}}$ with magnitude $|\lambda_1|$ and a maximal singular value $\sigma_1^{(l)}$ of $\boldsymbol{O}^{(l)}$:

$$\frac{1}{\left\| \frac{\boldsymbol{X}^{(l)}}{\lambda_1 \sigma_1^{(l)}} \right\|_F} \le \frac{1}{\left(\boldsymbol{N}_{:,1}^{(l)} \otimes \boldsymbol{P}_{:,1} \right)^\top \operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)} \tag{3.118}$$

Combining these bounds, we have

$$\left\| \frac{\boldsymbol{X}^{(l)}}{\|\boldsymbol{X}^{(l)}\|_F} - \boldsymbol{P}_{(1)}\boldsymbol{V}^{(l)} \right\|_F \le \frac{n \cdot \max_{i,j} P_{i,j}}{\left(\boldsymbol{N}_{:,1}^{(l)} \otimes \boldsymbol{P}_{:,1} \right)^\top \operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)} n \cdot l^{n-1} \cdot \left(\frac{|\lambda_2|}{|\lambda_1|} \right)^l \cdot b_{\max} \cdot \sqrt{n \cdot d} \tag{3.119}$$

Equivalently to the proof of Theorem 3.3, in order for this not to converge to zero, we need $\frac{1}{\left(\boldsymbol{N}^{(l)}_{:,1} \otimes \boldsymbol{P}_{:,1}\right)^\top \operatorname{vec}(\boldsymbol{X}^{(0)})}$ to be unbounded. The probability of this occurring over a bounded domain converges to zero. Thus, we have convergence in probability. □

Theorem 3.5 confirms that for any aggregation matrix $\tilde{\boldsymbol{A}}$ and any layer-dependent feature matrices $\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(l)}$, the node representations $\boldsymbol{X}^{(l)}$ converge to a low-rank state in probability for initial states $\boldsymbol{X}^{(0)}$. This rank is upper bounded by the number of eigenvalues of $\tilde{\boldsymbol{A}}$ with maximal magnitude. Based on this property, feature channels contain increasingly similar and redundant information with increased depth.

Rank Collapse and Over-Correlation

Analogously to connecting these properties for the special case of $\tilde{\boldsymbol{A}} = \boldsymbol{A}_{\text{sym}}$ to over-smoothing, we now generalize the concept of over-smoothing and relate it to over-correlation. Based on Theorem 3.5, we have seen that node representations get closer to a matrix whose rank is bounded by the number of eigenvalues of $\tilde{\boldsymbol{A}}$ with maximal magnitude. Over-smoothing is a consequence of $\boldsymbol{A}_{\text{sym}}$ having a unique eigenvalue with maximal magnitude, for which the corresponding eigenvector is $\boldsymbol{D}^{1/2}\boldsymbol{1}$. For the general case, we now equivalently assume that $\tilde{\boldsymbol{A}}$ has a unique eigenvalue with maximal magnitude, which holds for almost every $\tilde{\boldsymbol{A}}$ with respect to the Lebesgue measure (Griffiths et al. 1994). For such an aggregation matrix $\tilde{\boldsymbol{A}}$ with an arbitrary dominant eigenvector, this eigenvector will dominate every feature channel (i.e., column) of $\boldsymbol{X}^{(l)}$. More generally than converging to a specific rank-one matrix, we refer to the phenomenon of converging to any rank-one matrix, as rank collapse of node representations $\boldsymbol{X}^{(l)}$. We formalize this in the following proposition:

Proposition 3.9 (Rank Collapse in MPNNs). Let $\boldsymbol{X}^{(k)} = \tilde{\boldsymbol{A}}\boldsymbol{X}^{(k-1)}\boldsymbol{W}^{(k)}$, where $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d \times d}$ is any matrix for all $k \in \mathbb{N}$ and $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n \times n}$ is any matrix with $|\lambda_1| > |\lambda_j|$ for all $j > 1$ and corresponding eigenvector $\boldsymbol{u}$. Then,

$$\operatorname*{plim}_{l \to \infty} \left\| \frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F} - \boldsymbol{u}\left(\boldsymbol{v}^{(l)}\right)^\top \right\|_F = 0 \tag{3.120}$$

in probability for $\boldsymbol{X}^{(0)} \in [-m, m]^{n \times d}$ for any $m \in \mathbb{R}$ with respect to the Lebesgue measure and a layer-dependent $\boldsymbol{v}^{(l)}$ for all $l \in \mathbb{N}$.

Proof. Based on Theorem 3.5, we have

$$\operatorname*{plim}_{l \to \infty} \left\| \frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F} - \boldsymbol{P}_{(1)}\boldsymbol{V}^{(l)} \right\|_F = 0 \tag{3.121}$$

where $\boldsymbol{P}_{(1)}$ contains all generalized eigenvectors of $\tilde{\boldsymbol{A}}$ corresponding to eigenvalues with maximal magnitude. As we assume this eigenvalue λ_1 to be unique, $\boldsymbol{P}_{(1)}$ reduces to a matrix with a single column, which we denote by $\boldsymbol{u}$ and the corresponding matrix of coefficients $\boldsymbol{V}^{(l)}$ reduces to a matrix with a single row, which we denote as the transposed vector $\left(\boldsymbol{v}^{(l)}\right)^\top$. $\square$

As node representations approach a rank-one matrix, the feature channels become more similar and contain redundant information. The node representations of a rank-one matrix could be reduced to a single feature per node. Proposition 3.9 also confirms that the probability of having feature transformations $\boldsymbol{W}^{(k)}$ that prevent rank collapse converges to zero as the number of iterations goes to infinity. Thus, the effect of learnable parameters is limited for message-passing operations of this form.

Proposition 3.9 also explains the observed phenomenon of over-correlation. It was found that feature channels tend to become increasingly correlated with an increasing number of iterations, even when over-smoothing is not observed (Jin et al. 2022). Based on our findings above, when node representations are close to a rank-one matrix, the feature channels are also strongly correlated. Thus, over-correlation and over-smoothing describe similar properties of the underlying message-passing process.

As shown in Theorem 3.5, when the aggregation function $\tilde{\boldsymbol{A}}$ has multiple eigenvalues with maximal magnitude, the representation matrix still collapses to a low-rank subspace, though not necessarily of rank one. In this case, an over-correlation of feature channels may not be observed, but the underlying phenomenon still occurs.

3.5.3 The Effect of Iteration-Dependent Attention Weights

Various proposed message-passing methods utilize an iteration-dependent aggregation function $\tilde{\boldsymbol{A}}^{(k)}$ instead of a fixed aggregation function. Most of them use learnable attention coefficients, as employed in the graph attention network (GAT) (Velickovic et al. 2018), GATv2 (Brody et al. 2022), or graph Transformer (Shi et al. 2021; Rong, Bian, et al. 2020; Rampásek, Galkin, et al. 2022). The corresponding update rule is given as

$$\boldsymbol{X}^{(k)} = \tilde{\boldsymbol{A}}^{(k)}\boldsymbol{X}^{(k-1)}\boldsymbol{W}^{(k)} \tag{3.122}$$

where the attention coefficients of $\tilde{\boldsymbol{A}}^{(k)} \in \mathbb{R}^{n\times n}$ at iteration k depend on the features $\boldsymbol{X}^{(k-1)}$. We generally have $\tilde{\boldsymbol{A}}^{(i)} \neq \tilde{\boldsymbol{A}}^{(j)}$ for $i \neq j$. These are referred to as layer-dependent or time-inhomogeneous aggregation functions.

While the attention coefficients are iteration-dependent, they share some key properties. Attention coefficients are typically softmax-activated, meaning that the sum over each row is one, and all edge weights are non-negative. This corresponds to a node being updated using a weighted mean over its neighbors. Matrices $\tilde{\boldsymbol{A}}^{(k)}$ satisfying these properties are called row-stochastic. Such matrices have been extensively studied due to their importance in various fields (Gallager 1996). It is known that any row-stochastic matrix has the all-ones vector $\mathbf{1}$ as an eigenvector with corresponding eigenvalue $\lambda_1 = 1$. From the Perron-Frobenius theorem, it is also known that all other eigenvalues are smaller in magnitude, i.e., $|\lambda_i| < \lambda_1$ for all $i > 1$ (Gallager 1996).

For a single iteration, Proposition 3.7 applies and components of $\boldsymbol{X}^{(k-1)}$ are amplified across all columns depending on $\tilde{\boldsymbol{A}}^{(k)}$. However, the generalized eigenvectors and singular vectors are typically distinct across different row-stochastic matrices $\tilde{\boldsymbol{A}}^{(k)}$. Thus, our iterated findings cannot be directly applied.

We build on a result regarding the product of time-inhomogeneous aggregation matrices defined over a common ergodic graph (i.e., strongly connected and aperiodic) from Wu et al. (2023). They assume all $\tilde{\boldsymbol{A}}^{(k)}$ to have shared non-zero entries for all adjacent nodes, and each non-zero entry has a minimum value of $\epsilon > 0$. They found that the product of any such time-inhomogeneous aggregation matrices converges to the column-wise constant state, i.e., $\lim_{l\to\infty} \prod_{k=1}^{l} \tilde{\boldsymbol{A}}^{(k)} = \mathbf{1}\boldsymbol{u}^\top$ for some vector $\boldsymbol{u} \in \mathbb{R}^n$ indicating the constant value for each column.

This means that we can form two subspaces $\mathcal{Q}_1$ and $\mathcal{Q}_2$, with $\mathcal{Q}_1$ corresponding to the constant vector and the other subspace corresponding to all other components. For $\boldsymbol{X}^{(l)}$, the components corresponding to $\mathcal{Q}_1$ get increasingly amplified, while the components corresponding to $\mathcal{Q}_2$ become negligible. As the constant vector corresponds to the null space of the unnormalized graph Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$, we can show that the Dirichlet energy $E_{\boldsymbol{L}}(\boldsymbol{X}) = \operatorname{tr}\left(\boldsymbol{X}^\top \boldsymbol{L}\boldsymbol{X}\right)$ converges to zero for the normalized state $\frac{\boldsymbol{X}^{(l)}}{\|\boldsymbol{X}^{(l)}\|_F}$, independently of the chosen feature transformations:

Proposition 3.10 (Iteration-dependent attention exhibits over-smoothing). Let $\boldsymbol{X}^{(k)} = \tilde{\boldsymbol{A}}^{(k)}\boldsymbol{X}^{(k-1)}\boldsymbol{W}^{(k)}$ where each $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d\times d}$ is any matrix and $\tilde{\boldsymbol{A}}^{(k)} \in \mathbb{R}^{n\times n}$ is a row-stochastic matrix, representing the same ergodic graph with minimum non-zero value ϵ in $\tilde{\boldsymbol{A}}$ for some $\epsilon > 0$. Then,

$$\operatorname*{plim}_{l\to\infty} E_{\boldsymbol{L}}\left(\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}\right) = 0\,. \tag{3.123}$$

for $\boldsymbol{X}^{(0)} \in [-m, m]^{n\times d}$ for any $m \in \mathbb{R}$ in probability with respect to the Lebesgue measure.

Proof. Based on (Wu et al. 2023), we use the fact that

$$\lim_{l\to\infty} \tilde{\boldsymbol{A}}^{(l)} \dots \tilde{\boldsymbol{A}}^{(1)} = 1(\boldsymbol{y})^\top\,. \tag{3.124}$$

We utilize the induced decomposition

$$\tilde{\boldsymbol{A}}^{(l)} \dots \tilde{\boldsymbol{A}}^{(1)} = 1(\boldsymbol{y})^\top + \tilde{\boldsymbol{A}}_-^{(l)}\,, \tag{3.125}$$

where $\tilde{\boldsymbol{A}}_-^{(l)} = \tilde{\boldsymbol{A}}^{(l)} \dots \tilde{\boldsymbol{A}}^{(1)} - 1(\boldsymbol{y})^\top$ with $\left\|\tilde{\boldsymbol{A}}_-^{(l)}\right\|_2 \leq Cq^l$ for some $C \in \mathbb{R}$ and $0 < q < 1$. As $\mathbf{1}$ is in the nullspace of $\boldsymbol{L}$, i.e., $\boldsymbol{L}\mathbf{1} = \mathbf{0}$, we have

$$\left(\tilde{\boldsymbol{A}}^{(l)} \dots \tilde{\boldsymbol{A}}^{(1)}\right)^\top \boldsymbol{L}\left(\tilde{\boldsymbol{A}}^{(l)} \dots \tilde{\boldsymbol{A}}^{(1)}\right) = \left(\tilde{\boldsymbol{A}}_-^{(l)}\right)^\top \boldsymbol{L}\tilde{\boldsymbol{A}}_-^{(l)}\,. \tag{3.126}$$

Based on the definition of $E_{\boldsymbol{L}}$ and the property of the Kronecker product that $\operatorname{vec}(\boldsymbol{ABC}) = \left(\boldsymbol{C}^\top \otimes \boldsymbol{A}\right) \operatorname{vec}(\boldsymbol{B})$ for matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ of compatible shape, we obtain

$$E_{\boldsymbol{L}}\left(\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}\right) = \frac{\operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)^\top \left(\left(\boldsymbol{O}^{(l)}\right)^\top \boldsymbol{O}^{(l)} \otimes \left(\tilde{\boldsymbol{A}}_-^{(l)}\right)^\top \boldsymbol{L}\tilde{\boldsymbol{A}}_-^{(l)}\right) \operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)}{\left\|\boldsymbol{X}^{(l)}\right\|_F^2}, \tag{3.127}$$

where $\boldsymbol{O}^{(l)} = \boldsymbol{W}^{(1)} \dots \boldsymbol{W}^{(l)}$. By normalizing by the largest singular value $\sigma_1^{(l)}$ of $\boldsymbol{O}^{(l)}$, we obtain

$$E_{\boldsymbol{L}}\left(\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}\right) = \frac{\operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)^\top \left(\frac{\left(\boldsymbol{O}^{(l)}\right)^\top \boldsymbol{O}^{(l)}}{\left(\sigma_1^{(l)}\right)^2} \otimes \left(\tilde{\boldsymbol{A}}_-^{(l)}\right)^\top \left(\boldsymbol{L}\right) \tilde{\boldsymbol{A}}_-^{(l)}\right) \operatorname{vec}\left(\boldsymbol{X}^{(0)}\right))}{\frac{\left\|\boldsymbol{X}^{(l)}\right\|_F^2}{\left(\sigma_1^{(l)}\right)^2}} \tag{3.128}$$

Using the sub-multiplicativity, we upper-bound each term individually and obtain

$$E_{\boldsymbol{L}}\left(\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}\right) \leq \frac{\left\|\operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)\right\|_2^2 \left\|\frac{\boldsymbol{O}^{(l)}}{\sigma_1^{(l)}}\right\|_2^2 \left\|\boldsymbol{A}_-^{(l)}\right\|_2^2 \left\|\boldsymbol{L}\right\|_2}{\frac{\left\|\boldsymbol{X}^{(l)}\right\|_F^2}{\left(\sigma_1^{(l)}\right)^2}} \tag{3.129}$$

As $\lim_{l\to\infty} \left\|\tilde{\boldsymbol{A}}_-^{(l)}\right\|_2 \leq Cq^l = 0$, $\|\boldsymbol{L}\|_2 \leq 2 \cdot d_{\max}$ where $d_{\max}$ is the maximum degree of a node in the given graph, and $\left\|\frac{\boldsymbol{O}^{(l)}}{\sigma_1^{(l)}}\right\|_2 = 1$, the numerator can be upper bounded as

$$E_{\boldsymbol{L}}\left(\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}\right) \leq \frac{\left\|\boldsymbol{X}^{(0)}\right\|_F^2 \cdot 2 \cdot d_{\max} \cdot C^2 \cdot q^{2l}}{\frac{\left\|\boldsymbol{X}^{(l)}\right\|_F^2}{\left(\sigma_1^{(l)}\right)^2}}. \tag{3.130}$$

Next, we bound the denominator analogously to the proof of Theorem 3.3 to obtain

$$\frac{1}{\frac{\left\|\boldsymbol{X}^{(l)}\right\|_F^2}{\left(\sigma_1^{(l)}\right)^2}} \geq \frac{1}{\left(\left(\boldsymbol{N}_{:,1}^{(l)} \otimes \boldsymbol{1}\right)^\top \operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)\right)^2}, \tag{3.131}$$

where $\boldsymbol{O}^{(l)} = \boldsymbol{V}^{(l)}\boldsymbol{\Sigma}^{(l)}\left(\boldsymbol{N}^{(l)}\right)^\top$ is the singular value decomposition of $\boldsymbol{O}^{(l)}$ and $\boldsymbol{N}_{:,1}^{(l)} \in \mathbb{R}^{d\times 1}$ contains the right singular vector associated to $\sigma_1^{(l)}$ as its sole column.

Combining these upper bounds, we obtain

$$E_{\boldsymbol{L}}\left(\frac{\boldsymbol{X}^{(l)}}{\left\|\boldsymbol{X}^{(l)}\right\|_F}\right) \leq \frac{\left\|\boldsymbol{X}^{(0)}\right\|_F^2 \cdot 2 \cdot d_{\max} \cdot C^2 \cdot q^{2l}}{\left(\left(\boldsymbol{N}_{:,1}^{(l)} \otimes \boldsymbol{1}\right)^\top \operatorname{vec}\left(\boldsymbol{X}^{(0)}\right)\right)^2} \tag{3.132}$$

As the numerator converges to zero exponentially, $\boldsymbol{X}^{(0)}$ needs to lie in an increasingly small subspace to avoid convergence to zero of the complete term. Since the measure of this subspace shrinks exponentially, the probability of randomly observing such an $\boldsymbol{X}^{(0)}$ from a bounded set converges to zero. $\square$

Proposition 3.10 confirms that our insights also hold for attention-based aggregation functions. Repeatedly applying such update steps leads to the dominance of the constant vector across all feature channels. As this constant vector is in the null space of the unnormalized graph Laplacian $\boldsymbol{L}$, the Dirichlet energy converges to zero for initial representations $\boldsymbol{X}^{(0)}$ with probability converging to one when $l \to \infty$. This finding can be similarly applied to any iteration-dependent aggregation functions $\tilde{\boldsymbol{A}}^{(1)}, \dots \tilde{\boldsymbol{A}}^{(l)}$ whenever their product dominates certain components. However, these theoretical insights do not extend to the multi-head case, where h attention matrices $\boldsymbol{A}_1^{(k)}, \dots, \boldsymbol{A}_h^{(k)}$ are used for each iteration k.

3.5.4 Graph Convolutions as Power Iteration

To provide further intuition, we now show that the limit behavior of $\frac{\boldsymbol{X}^{(l)}}{\|\boldsymbol{X}^{(l)}\|_F}$ for $l \to \infty$ is a special case of the classical power iteration algorithm (Kowalewski 1909; Mises et al. 1929) when using a shared feature transformation, i.e., $\boldsymbol{W}^{(1)} = \dots = \boldsymbol{W}^{(l)}$. This connection to a well-established algorithm and behavior will deepen our understanding of the over-smoothing and rank collapse phenomena. We will first recap the power iteration method and its general proof, before showing that many variations of graph convolutions can be seen as a special case of power iteration. Thus, these graph convolutions exhibit the same properties and standard textbook proofs can be applied. This subsection is based on (Roth 2024).

Power Iteration

The power method or power iteration is an algorithm to determine the dominant eigenvector and eigenvalue of a matrix $\boldsymbol{T} \in \mathbb{R}^{p \times p}$. The key property used within power iteration is that repeatedly applying a matrix $\boldsymbol{T}$ to a vector $\boldsymbol{x} \in \mathbb{R}^p$ leads to the resulting vector $\boldsymbol{T}^l\boldsymbol{x}$ getting increasingly dominated by the eigenvector $\boldsymbol{U}_{:,1}$ of $\boldsymbol{T}$ corresponding to its eigenvalue with maximal magnitude. This holds for almost every matrix $\boldsymbol{T}$, as it has a single eigenvalue with maximal magnitude, and almost every $\boldsymbol{x}$, as its component corresponding to $\boldsymbol{U}_{:,1}$ needs to be non-zero. This means that $\frac{\boldsymbol{T}^k\boldsymbol{x}}{\|\boldsymbol{T}^k\boldsymbol{x}\|_2}$ becomes increasingly equal to plus or minus $\boldsymbol{U}_{:,1}$. We will first present a commonly used proof for power iteration in general before relating this to graph convolutions and MPNNs:

Proposition 3.11 (Power Iteration (Knabner et al. 2017)). Let $\boldsymbol{T} \in \mathbb{R}^{p \times p}$ for some $p \in \mathbb{N}$ with an eigenvalue $|\lambda_1| > |\lambda_i|$ that is greater in magnitude than all other eigenvalues.

Further, let $\boldsymbol{U}_{:,1} \in \mathbb{R}^p$ be a normalized eigenvector of $\boldsymbol{T}$ corresponding to λ_1. Then, for almost every $\boldsymbol{x} \in \mathbb{R}^p$,

$$\frac{\boldsymbol{T}^k \boldsymbol{x}}{\left\|\boldsymbol{T}^k \boldsymbol{x}\right\|_2} = \beta_k \boldsymbol{U}_{:,1} + \boldsymbol{m}^{(k)} \tag{3.133}$$

for $\boldsymbol{m}^{(k)} \in \mathbb{R}^p$ with $\lim_{k\to\infty} \left\|\boldsymbol{m}^{(k)}\right\|_2 = 0$ and $\beta_k \in \{-1, 1\}$ for every $k \in \mathbb{N}$.

Proof. The proof utilizes the Jordan normal form $\boldsymbol{T} = \boldsymbol{P}\boldsymbol{J}\boldsymbol{P}^{-1}$. As $\boldsymbol{P}$ contains the generalized eigenvectors of $\boldsymbol{T}$, we have $\boldsymbol{U}_{:,1} = \boldsymbol{P}_{:,1}$ and $J_{1,1} = \lambda_1$. This simplifies the k-th power of $\boldsymbol{T}$ to $\boldsymbol{T}^k = \boldsymbol{P}\boldsymbol{J}^k\boldsymbol{P}^{-1}$. We also decompose $\boldsymbol{x}$ into a linear combination of $\boldsymbol{P}$ by setting $\boldsymbol{c} = \boldsymbol{P}^{-1}\boldsymbol{x}$. Based on these two properties, we have

$$\frac{\boldsymbol{T}^k \boldsymbol{x}}{\left\|\boldsymbol{T}^k \boldsymbol{x}\right\|_2} = \frac{\boldsymbol{P}\boldsymbol{J}^k \boldsymbol{c}}{\left\|\boldsymbol{P}\boldsymbol{J}^k \boldsymbol{c}\right\|_2}. \tag{3.134}$$

By normalizing the numerator and denominator by $|\lambda_1|$ and the corresponding entry c_1 of $\boldsymbol{c}$, we obtain

$$\frac{\boldsymbol{T}^k \boldsymbol{x}}{\left\|\boldsymbol{T}^k \boldsymbol{x}\right\|_2} = \frac{c_1}{|c_1|}\left(\frac{|\lambda_1|}{|\lambda_1|}\right)^k \frac{\boldsymbol{P}\left(\frac{\boldsymbol{J}}{|\lambda_1|}\right)^k \frac{\boldsymbol{c}}{c_1}}{\left\|\boldsymbol{P}\left(\frac{\boldsymbol{J}}{|\lambda_1|}\right)^k \frac{\boldsymbol{c}}{|c_1|}\right\|_2}. \tag{3.135}$$

As $\boldsymbol{J}$ is a block diagonal matrix, which has λ_1 as its maximal entry $J_{1,1}$ on its diagonal with a corresponding block size of 1, normalizing by $|\lambda_1|$ results in all other entries converging to zero, i.e., we have

$$\lim_{k\to\infty}\left(\frac{\boldsymbol{J}}{|\lambda_1|}\right)^k = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & 0 \end{bmatrix} \tag{3.136}$$

Thus, all other components c_i for $i \neq 1$ also converge to zero, and we obtain

$$\frac{\boldsymbol{T}^k \boldsymbol{x}}{\left\|\boldsymbol{T}^k \boldsymbol{x}\right\|_2} = \frac{c_1}{|c_1|}\left(\frac{|\lambda_1|}{|\lambda_1|}\right)^k \frac{\boldsymbol{P}_{:,1}}{\left\|\boldsymbol{P}_{:,1}\right\|_2} + \boldsymbol{m}^{(k)} \tag{3.137}$$

for some $\boldsymbol{m}^{(k)}$ with $\lim_{k\to\infty} \left\|\boldsymbol{m}^{(k)}\right\|_2 = 0$. We further have $\frac{c_1|\lambda_1^k|}{|c_1|\cdot|\lambda_1^k|} \in \{-1, 1\}$. When the eigenvalue with maximal magnitude is positive, i.e., $\lambda_1 > 0$, the state $\frac{\boldsymbol{T}^k \boldsymbol{x}}{\left\|\boldsymbol{T}^k \boldsymbol{x}\right\|_2}$ converges. Otherwise, it oscillates between the positive and negative normalized eigenvector but still converges in direction to $\boldsymbol{P}_{:,1}$.

This proof requires $c_1 \neq 0$, which is satisfied for almost every $\boldsymbol{x} \in \mathbb{R}^p$ with respect to the Lebesgue measure on $\mathbb{R}^p$. □

We summarize this property as follows: When repeatedly applying a matrix $\boldsymbol{T}$ to a vector, the eigenvector of $\boldsymbol{T}$ corresponding to the eigenvalue of $\boldsymbol{T}$ with maximal magnitude increasingly dominates the resulting vector. The components corresponding to all other directions becomes comparably negligible. The condition that $\boldsymbol{T}$ has a unique eigenvalue $|\lambda_1| > |\lambda_i|$ with maximal magnitude is satisfied for almost every matrix $\boldsymbol{T}$ with respect to the Lebesgue measure (Tao 2012).

Graph Convolutions as Power Iteration

Many variations of graph convolutions or message-passing iterations can be similarly expressed as a matrix-vector product

$$\operatorname{vec}\left(\boldsymbol{X}^{(k)}\right) = \operatorname{vec}\left(\tilde{\boldsymbol{A}}\boldsymbol{X}^{(k-1)}\boldsymbol{W}\right) = \left(\boldsymbol{W}^\top \otimes \tilde{\boldsymbol{A}}\right)^k \operatorname{vec}\left(\boldsymbol{X}^{(0)}\right) = \boldsymbol{S}^k \boldsymbol{x}^{(0)} \tag{3.138}$$

where $\boldsymbol{S} = \left(\boldsymbol{W}^\top \otimes \tilde{\boldsymbol{A}}\right) \in \mathbb{R}^{nd \times nd}$ and $\boldsymbol{x}^{(0)} = \operatorname{vec}\left(\boldsymbol{X}^{(k)}\right) \in \mathbb{R}^{nd}$. Proposition 3.11 directly applies to Equation (3.138) under the assumption that $\boldsymbol{S} = \boldsymbol{W}^\top \otimes \tilde{\boldsymbol{A}}$ has a unique eigenvalue with maximal magnitude. Based on power iteration, vec $\left(\boldsymbol{X}^{(k)}\right)$ results in a state that is dominated by the corresponding eigenvector of $\boldsymbol{S}$.

While this applies to any linear operator $\boldsymbol{S}$, the special form of $\boldsymbol{S}$ in Equation (3.138) allows us to define the property more precisely. As $\boldsymbol{S} = \boldsymbol{W}^\top \otimes \tilde{\boldsymbol{A}}$ is a Kronecker product of two matrices, its eigenvalues and eigenvectors have a specific form. In the following, we denote $\lambda_i^{(\boldsymbol{M})}$ as the i-th largest eigenvalue in magnitude of $\boldsymbol{M}$ and $\boldsymbol{U}_{:,i}^{(\boldsymbol{M})}$ as the corresponding eigenvector of matrix $\boldsymbol{M}$. Based on the mixed-product property of the Kronecker product and the matrix product, each eigenvector $\boldsymbol{U}_{:,m}$ of $\boldsymbol{S}$ is a Kronecker products of the eigenvectors of $\boldsymbol{W}^\top$ and $\tilde{\boldsymbol{A}}$, i.e., the eigenvectors are given by $\boldsymbol{U}_{:,(i-1)n+j}^{(\boldsymbol{S})} = \boldsymbol{U}_{:,i}^{(\boldsymbol{W}^\top)} \otimes \boldsymbol{U}_{:,j}^{(\tilde{\boldsymbol{A}})}$ for $i \in [d]$ and $j \in [n]$ (Horn et al. 1991). The corresponding eigenvalue is given by the product of eigenvalues $\lambda_i^{(\boldsymbol{W}^\top)}$ of $\boldsymbol{W}^\top$ and $\lambda_j^{(\tilde{\boldsymbol{A}})}$ of $\tilde{\boldsymbol{A}}$ as $\lambda_{:,(i-1)n+j}^{(\boldsymbol{S})} = \lambda_i^{(\boldsymbol{W}^\top)} \lambda_j^{\tilde{\boldsymbol{A}}}$ for $i \in [d]$ and $j \in [n]$ (Horn et al. 1991). Based on these additional properties of the considered $\boldsymbol{S}$, power iteration shows that repeatedly applying such graph convolutions results in vec$(\boldsymbol{X}^{(k)})$ becoming increasingly dominated by $\boldsymbol{U}_{:,1}^{(\boldsymbol{S})} = \boldsymbol{U}_{:,1}^{(\boldsymbol{W}^\top)} \otimes \boldsymbol{U}_{:,1}^{(\tilde{\boldsymbol{A}})}$. While the insight by this Proposition is similar to Theorem 3.5 and Theorem 3.5, the relation to the well-known power iteration method gives it additional understanding and intuition. We formalize this property in the following corollary:

Corollary 3.2 (Power Iteration with a Kronecker Product). Let $\boldsymbol{S} = \boldsymbol{W}^\top \otimes \tilde{\boldsymbol{A}} \in \mathbb{R}^{nd \times nd}$ for any $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ and $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n \times n}$ with an eigenvalue $|\lambda_1^{(\boldsymbol{S})}| > |\lambda_i^{(\boldsymbol{S})}|$ that is greater in magnitude than all other eigenvalues. Then, for almost every $\boldsymbol{x}^{(0)}$,

$$\frac{\boldsymbol{S}^k \boldsymbol{x}^{(0)}}{\left\|\boldsymbol{S}^k \boldsymbol{x}^{(0)}\right\|_2} = \beta_k \left(\boldsymbol{U}_{:,1}^{(\boldsymbol{W}^\top)} \otimes \boldsymbol{U}_{:,1}^{(\tilde{\boldsymbol{A}})}\right) + \boldsymbol{m}^{(k)} \tag{3.139}$$

for $\boldsymbol{m}^{(k)}$ with $\lim_{k\to\infty} \left\|\boldsymbol{m}^{(k)}\right\|_2 = 0$ and $\beta_k \in \{-1, 1\}$ for every $k \in \mathbb{N}$.

Proof. Based on Theorem 3.11, we know that

$$\frac{\boldsymbol{S}^k \boldsymbol{x}^{(0)}}{\left\|\boldsymbol{S}^k \boldsymbol{x}^{(0)}\right\|_2} = \beta_k \boldsymbol{U}_{:,1}^{(\boldsymbol{S})} + \boldsymbol{m}^{(k)} \tag{3.140}$$

for $\beta_k \in \{-1, 1\}$ and some $\boldsymbol{m}^{(k)}$ with $\lim_{k\to\infty} \left\|\boldsymbol{m}^{(k)}\right\|_2 = 0$. Based on the mixed-product property of the Kronecker product and the matrix product, the eigenvector $\boldsymbol{U}_{:,1}^{(\boldsymbol{S})}$ is given by the Kronecker product $\boldsymbol{U}_{:,1}^{(\boldsymbol{S})} = \boldsymbol{U}_{:,1}^{(\boldsymbol{W}^\top)} \otimes \boldsymbol{U}_{:,1}^{(\tilde{\boldsymbol{A}})}$ of eigenvectors of $\tilde{\boldsymbol{A}}$ and $\boldsymbol{W}^\top$. Substituting the definition of eigenvector $\boldsymbol{U}_{:,1}^{(\boldsymbol{S})}$ concludes this corollary. □

While Corollary 3.2 already establishes the connection to our previous findings, we additionally state Corollary 3.2 in matrix notation to further highlight the equivalence:

Corollary 3.3 (Power Iteration with a Kronecker Product). Let $\boldsymbol{X}^{(k)} = \tilde{\boldsymbol{A}}\boldsymbol{X}^{(k-1)}\boldsymbol{W}$ for $\boldsymbol{W} \in \mathbb{R}^{d\times d}$ and $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n\times n}$ matrices with eigenvalues $|\lambda_1^{(\boldsymbol{W})}| > |\lambda_i^{(\boldsymbol{W})}|$ and $|\lambda_1^{(\tilde{\boldsymbol{A}})}| > |\lambda_i^{(\tilde{\boldsymbol{A}})}|$ that are greater in magnitude than all other eigenvalues of $\boldsymbol{W}$ and $\tilde{\boldsymbol{A}}$, respectively. Then,

$$\frac{\boldsymbol{X}^{(k)}}{\left\|\boldsymbol{X}^{(k)}\right\|_F} = \beta_k \boldsymbol{U}_{:,1}^{(\tilde{\boldsymbol{A}})} \left(\boldsymbol{U}_{:,1}^{(\boldsymbol{W})}\right)^\top + \boldsymbol{M}^{(k)} \tag{3.141}$$

for almost every $\boldsymbol{X}^{(0)} \in \mathbb{R}^{n\times d}$ and $\beta_k \in \{-1, 1\}$ and $\boldsymbol{M}^{(k)} \in \mathbb{R}^{n\times d}$ for every $k \in \mathbb{N}$ with $\lim_{l\to\infty} \left\|\boldsymbol{M}^{(l)}\right\|_2 = 0$.

Despite Corollary 3.2 and Corollary 3.3 considering a simplified case of graph convolutions and message-passing methods, they highlight the underlying property behind over-smoothing and more general phenomena. As the aggregation function $\tilde{\boldsymbol{A}}$ is fixed, its dominant eigenvector $\boldsymbol{U}_{:,1}^{(\tilde{\boldsymbol{A}}}$ is also predefined. The dominance of this vector across each feature channel cannot be prevented by the feature transformation $\boldsymbol{W}$. Thus, the learnable parameters given by $\boldsymbol{W}$ have a limited effect on the resulting representations. As feature channels become increasingly similar and contain more redundant information, additional feature channels and parameters similarly have a limited effect. For specific choices of the aggregation function $\tilde{\boldsymbol{A}}$, we know in advance to which vector each feature channel becomes more similar with increased depth. For $\boldsymbol{A}_{\text{sym}}$, we have $\boldsymbol{U}_{:,1} = \boldsymbol{D}^{1/2}\boldsymbol{1}$, resulting in each feature channel becoming increasingly similar to this degree-proportional state. For $\boldsymbol{A}_{\text{rw}}$ or other row-stochastic matrices, we have $\boldsymbol{U}^{(\boldsymbol{A}_{\text{rw}})} = \boldsymbol{1}$, resulting in each feature channel becoming increasingly similar to the constant vector. As such, over-smoothing can be observed as a special case of this phenomenon.

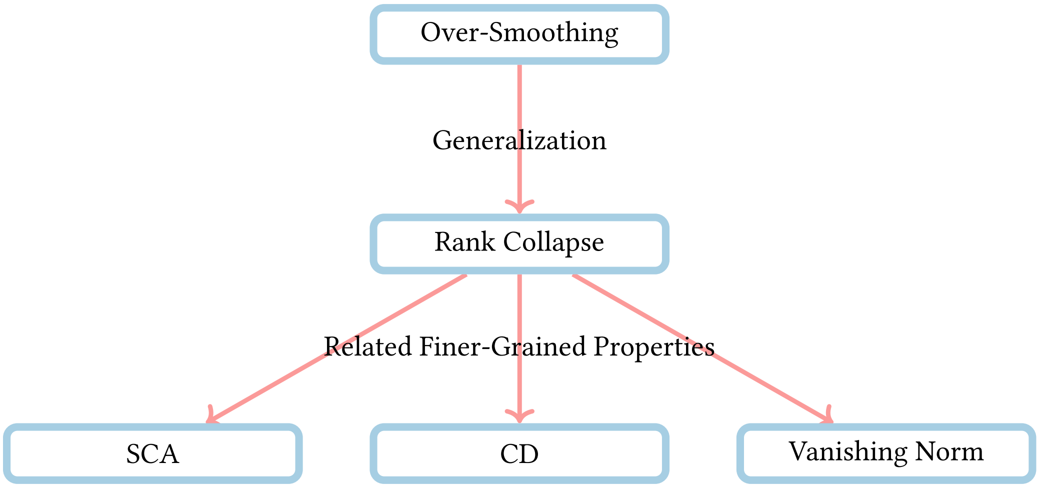


Figure 3.11: The connections between our identified phenomena and properties.

3.6 Summary

We now summarize our extensive findings from this section, introduce formal definitions, and relate our findings to a broader context. While many variations of MPNNs exist, we consider iterative update functions, for which each iteration k constructs an updated representation for each node v_i by computing

$$\boldsymbol{X}_{i,:}^{(k)} = \sum_{v_j \in N_i} \tilde{A}_{i,j} \boldsymbol{X}_{j,:}^{(k-1)} \boldsymbol{W}^{(k)} , \tag{3.142}$$

where $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n \times n}$ encodes the graph structure and the utilized aggregation function as edge weights, $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d \times d}$ is an iteration-dependent feature transformation, and $\boldsymbol{X}^{(k)} \in \mathbb{R}^{n \times d}$ is the matrix of node representations after iteration k. This form can represent various established methods, e.g., when the edge weights represent the mean or the sum aggregation. The edge weights may also be obtained by a more complex procedure, e.g., as a function of the node representations. We consider linearized iterations, in which we omit the activation function.

Based on our theoretical findings, we first define two properties that we have shown to hold for all message-passing methods of the form given in Equation (3.142). These are shared component amplification (SCA), which occurs for each individual message-passing iteration, and component dominance, a phenomenon that occurs with increased depth. We then relate these properties to the observable phenomenon of rank collapse of node representations, with over-smoothing as a special case of rank collapse. We also propose to consider our identified vanishing norm phenomenon as a separate challenge. These connections are visualized in Figure 3.11. Based on our findings, we also introduce the rank-one distance as a novel metric to quantify the rank collapse of node representations. While we show that all message-passing methods of the form given by Equation (3.142) exhibit SCA and CD, we also introduce a message-passing framework that does not have these properties.

For notational clarity, we will summarize our findings using an equivalent matrix notation of Equation (3.142) as

$$\boldsymbol{X}^{(k)} = \tilde{\boldsymbol{A}}\boldsymbol{X}^{(k-1)}\boldsymbol{W}^{(k)} . \tag{3.143}$$

For further notational simplicity, we use the equivalent vectorized form

$$\operatorname{vec}\left(\boldsymbol{X}^{(k)}\right) = \boldsymbol{T}^{(k)} \operatorname{vec}\left(\boldsymbol{X}^{(k-1)}\right) , \tag{3.144}$$

where $\boldsymbol{T}^{(k)} = \left(\boldsymbol{W}^{(k)}\right)^{\top} \otimes \tilde{\boldsymbol{A}} \in \mathbb{R}^{nd \times nd}$ is the Kronecker product $\otimes$ of matrices $\left(\boldsymbol{W}^{(k)}\right)^{\top}$ and $\tilde{\boldsymbol{A}}$. We start by defining SCA.

3.6.1 Shared Component Amplification

Following our findings from Theorem 3.2, Proposition 3.7, and Theorem 3.4, we define SCA as follows:

Definition 3.1 (Shared Component Amplification (SCA)). *Let $\mathcal{M}$ be a set of matrices with $\boldsymbol{M} \in \mathcal{M}$ satisfying $\boldsymbol{M} \in \mathbb{R}^{nd \times nd}$. We say that $\mathcal{M}$ exhibits **shared component amplification (SCA)** if there exist vectors $\boldsymbol{p}_{(i)} \in \mathbb{R}^n$ for $i \in [n]$ where the set $\{\boldsymbol{p}_{(1)}, \dots, \boldsymbol{p}_{(n)}\}$ spans $\mathbb{R}^n$, and there exist constants $a_1, \dots, a_n \in \mathbb{R}$ satisfying*

$$\frac{\left\|\boldsymbol{M}\left(\boldsymbol{I}_d \otimes \boldsymbol{p}_{(i)}\right)\right\|_F}{\left\|\boldsymbol{M}\left(\boldsymbol{I}_d \otimes \boldsymbol{p}_{(j)}\right)\right\|_F} = \frac{a_i}{a_j} \tag{3.145}$$

for all $i, j \in [n]$ and all matrices $\boldsymbol{M} \in \mathcal{M}$.

To provide an informal explanation for this definition, we first note that by construction of matrices $\boldsymbol{I}_d \otimes \boldsymbol{p}_{(i)} \in \mathbb{R}^{nd \times d}$, any linear combination $\operatorname{vec}(\boldsymbol{X}) = \left(\boldsymbol{I}_d \otimes \boldsymbol{p}_{(i)}\right)\boldsymbol{c}$ for $\boldsymbol{c} \in \mathbb{R}^d$ induces a rank-one matrix $\boldsymbol{X}$. Thus, when the set of matrices $\mathcal{M}$ satisfies the SCA property, components corresponding to $\boldsymbol{p}_{(i)}$ are amplified by a fixed relative amount across all feature channels, independently of the matrix $\boldsymbol{M} \in \mathcal{M}$. Thus, it is not possible to amplify one component maximally for one of the feature channels while amplifying another component maximally for a different feature channel. Based on Proposition 3.7, any iteration of the form given by Equation (3.142) with an arbitrary aggregation matrix $\tilde{\boldsymbol{A}}$ and all linear feature transformation $\boldsymbol{W}$ exhibits SCA, which we state here to align with Definition 3.1:

Theorem 3.6 (Shared Component Amplification). Let $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n \times n}$ be any matrix and $\mathcal{M} = \left\{ \boldsymbol{W} \otimes \tilde{\boldsymbol{A}} \mid \boldsymbol{W} \in \mathbb{R}^{d \times d} \right\}$ be a set of matrices. Then, $\mathcal{M}$ exhibits shared component amplification.

As Theorem 3.6 is equivalent to Proposition 3.7, we refer to the proof provided there. Central to our proof is the property of a matrix $\boldsymbol{M} = \boldsymbol{W} \otimes \tilde{\boldsymbol{A}}$ that each eigenvector $\boldsymbol{u}^{(\boldsymbol{M})}$ of $\boldsymbol{M}$ is a Kronecker product $\boldsymbol{u}^{(\boldsymbol{M})} = \boldsymbol{u}^{(\boldsymbol{W})} \otimes \boldsymbol{u}^{(\tilde{\boldsymbol{A}})}$ of eigenvectors $\boldsymbol{u}^{(\boldsymbol{W})}$ of $\boldsymbol{W}$ and $\boldsymbol{u}^{(\tilde{\boldsymbol{A}})}$ of $\tilde{\boldsymbol{A}}$. Such eigenvectors are vectorized rank-one matrices $\boldsymbol{u}^{(\boldsymbol{M})} = \operatorname{vec}\left(\boldsymbol{u}^{(\tilde{\boldsymbol{A}})} \left(\boldsymbol{u}^{(\boldsymbol{W})}\right)^\top\right)$. For singular vectors of $\boldsymbol{M}$, this is equivalent. As $\boldsymbol{M}$ amplifies components corresponding to eigenvectors and singular vectors, $\boldsymbol{M}$ amplifies these components sharedly across all feature channels.

This confirms that whenever a message-passing iteration of the form given by Equation (3.142) is applied, components get amplified with a fixed ratio for all feature channels and all feature transformation $\boldsymbol{W}$. This severely limits the potential of such MPNNs, as more than one feature channel leads to redundancy. With Theorem 3.2, we considered the special case of setting the aggregation matrix $\tilde{\boldsymbol{A}} = \boldsymbol{A}_{\text{sym}}$ as the symmetrically normalized adjacency matrix $\boldsymbol{A}_{\text{sym}}$. In this case, the eigenvectors of $\boldsymbol{A}_{\text{sym}}$ can be selected as vectors $\boldsymbol{p}_{(i)}$, and the components corresponding to each eigenvector get relatively amplified by the ratio of their respective eigenvalues. Consequently, the eigenvector $\boldsymbol{U}_{:,1} = \boldsymbol{D}^{1/2}\mathbf{1}$ gets maximally amplified for every feature channel, resulting in a smoothing of representations. Based on Theorem 3.4, we found that for arbitrary aggregation matrices $\tilde{\boldsymbol{A}}$, the right singular vectors can be used as vectors $\boldsymbol{p}_{(i)}$. Thus, the component of the representation matrix corresponding to the maximal singular value is maximally amplified across all feature channels and for all learnable parameters.

Thus, MPNNs of the form given in Equation (3.142) are inherently limited. A single message-passing iteration cannot amplify distinct components across feature channels, which holds true for any choice of learnable parameters $\boldsymbol{W}$. This also limits the effect of the optimization process and of additional parameters or feature channels.

3.6.2 Component Dominance

In addition to SCA occurring for each individual message-passing iteration of the form given in Equation (3.142), we also identified a related property for repeatedly applying such iterations. Based on our findings from Proposition 3.4, Corollary 3.1, and Proposition 3.8, we define CD as follows:

Definition 3.2 (Component Dominance (CD)). *A sequence of matrices $\left(\boldsymbol{M}^{(i)}\right)_{i\in\mathbb{N}}$, satisfying $\boldsymbol{M}^{(k)} \in \mathbb{R}^{m\times m}$ for any $m \in \mathbb{N}$, exhibits* ***component dominance (CD)*** *if there exist matrices $\boldsymbol{P}_{(1)} \in \mathbb{R}^{m\times m_1}, \dots, \boldsymbol{P}_{(p)} \in \mathbb{R}^{m\times m_p}$ with $p > 1$ and $m_1 + \dots + m_p = m$ such that the columns of $\boldsymbol{P}_{(1)}, \dots, \boldsymbol{P}_{(p)}$ together span $\mathbb{R}^m$ and*

$$\lim_{l\to\infty} \frac{\left\|\boldsymbol{M}^{(l)} \dots \boldsymbol{M}^{(1)} \boldsymbol{P}_{(i)}\right\|_F}{\left\|\boldsymbol{M}^{(l)} \dots \boldsymbol{M}^{(1)} \boldsymbol{P}_{(1)}\right\|_F} = \begin{cases} 1 & \text{if } i = 1, \\ 0 & \text{if } i \neq 1 \end{cases} \tag{3.146}$$

for all $i \in [p]$.

When a sequence of transformations satisfies this property, the component corresponding to $\boldsymbol{P}_{(1)}$ gets increasingly amplified compared to the amplification of all other components. We found that any MPNN of the form given in Equation (3.142) exhibits CD:

Theorem 3.7 (Component Dominance). Let $\left(\boldsymbol{M}^{(i)}\right)_{i\in\mathbb{N}}$ be a sequence of matrices of the form

$$\boldsymbol{M}^{(i)} = \left(\boldsymbol{W}^{(i)}\right)^{\top} \otimes \tilde{\boldsymbol{A}},$$

where each $\boldsymbol{W}^{(i)} \in \mathbb{R}^{d\times d}$ and $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n\times n}$ is a fixed matrix whose eigenvalues do not all have the same magnitude. Then, the sequence $\left(\boldsymbol{M}^{(i)}\right)_{i\in\mathbb{N}}$ exhibits component dominance.

Theorem 3.7 is equivalent to Proposition 3.8, whose proof we refer to for further details. Our finding confirms that MPNNs of the form given in Equation (3.142) dominantly amplify a single component while relatively filtering out all other components. When using $\boldsymbol{A}_{\text{sym}}$ as the aggregation matrix, we have shown in Proposition 3.4 and Corollary 3.1 that the component corresponding to the dominant eigenvector $\boldsymbol{D}^{1/2}\mathbf{1}$ of $\boldsymbol{A}_{\text{sym}}$ gets dominantly amplified across all feature channels. As shown in Proposition 3.8, for an arbitrary aggregation matrix $\tilde{\boldsymbol{A}}$, $\boldsymbol{P}_{(1)}$ is of the form $\boldsymbol{P}_{(1)} = \boldsymbol{I}_d \otimes \boldsymbol{U}_{(1)}$ where $\boldsymbol{U}_{(1)}$ contains all generalized eigenvectors of $\tilde{\boldsymbol{A}}$ with an eigenvalue of maximal magnitude. Due to the Kronecker product, this dominant amplification of eigenvectors $\boldsymbol{U}_{(1)}$ is shared across all feature channels, similarly to SCA. In Section 3.5.4, we have related this phenomenon to the classical power iteration method (Kowalewski 1909; Mises et al. 1929). From power iteration, it is similarly known that repeatedly applying a matrix $\boldsymbol{M}$ to a vector results in the dominance of the component corresponding to the dominant eigenvector of $\boldsymbol{M}$. In our case, all eigenvectors of $\boldsymbol{M}$ are given as a Kronecker product of eigenvectors of $\boldsymbol{W}$ and $\tilde{\boldsymbol{A}}$.

3.6.3 Rank Collapse

With SCA and CD, we describe properties of any message-passing method of the form given in Equation (3.142) utilizing any aggregation matrix $\tilde{\boldsymbol{A}}$, which may represent any graph structure. We do not directly observe the identified amplification of components, rather we observe node representations $\boldsymbol{X}^{(k)}$ that get iteratively updated by these message-passing functions. However, SCA and CD have direct consequences on these node representations. When the message-passing steps satisfy both the SCA and CD properties, the dominantly amplified component identified by CD is constrained by the maximally amplified subspace induced by SCA. As a consequence, node representations $\boldsymbol{X}^{(k)}$ converge to a low-rank subspace, following our findings from Proposition 3.5, Theorem 3.5, Proposition 3.9. We define rank collapse as the case when node representations $\boldsymbol{X}^{(k)}$ become increasingly close to a rank-one state as the number of iterations k increases:

Definition 3.3 (Rank Collapse). *A sequence of matrices* $\left(\boldsymbol{X}^{(k)}\right)_{k\in\mathbb{N}}$ *exhibits **rank collapse** if there exists a sequence of rank-one matrices* $\left(\boldsymbol{Y}^{(k)}\right)_{k\in\mathbb{N}}$ *such that*

$$\lim_{k\to\infty}\left\|\frac{\boldsymbol{X}^{(k)}}{\left\|\boldsymbol{X}^{(k)}\right\|_F}-\boldsymbol{Y}^{(k)}\right\|_F=0\,. \tag{3.147}$$

With Proposition 3.9, we have shown that rank collapse occurs for any message-passing method given by Equation (3.142) under slight assumptions:

Theorem 3.8 (Rank collapse in probability for MPNNs.). Let $\boldsymbol{X}^{(k)}=\tilde{\boldsymbol{A}}\boldsymbol{X}^{(k-1)}\boldsymbol{W}^{(k)}$ for any matrix $\tilde{\boldsymbol{A}}\in\mathbb{R}^{n\times n}$ with a dominant eigenvalue $|\lambda_1|>|\lambda_i|$ for all $i>1$, and for a sequence of matrices $\left(\boldsymbol{W}^{(k)}\right)_{k\in\mathbb{N}}$ with $\boldsymbol{W}^{(k)}\in\mathbb{R}^{d\times d}$. Then, for any initial $\boldsymbol{X}^{(0)}\in[-m,m]^{n\times d}$, where $m\in\mathbb{R}$, the sequence $\left(\boldsymbol{X}^{(k)}\right)_{k\in\mathbb{N}}$ exhibits rank collapse in probability with respect to the Lebesgue measure over $\boldsymbol{X}^{(0)}$. Moreover, the rank-one matrices $\boldsymbol{Y}^{(k)}$ are of the form $\boldsymbol{Y}^{(k)}=\boldsymbol{U}_{:,1}\boldsymbol{n}_{(k)}^{\top}$, where $\boldsymbol{U}_{:,1}$ is the eigenvector of $\tilde{\boldsymbol{A}}$ corresponding to eigenvalue λ_1, and $\boldsymbol{n}_{(k)}\in\mathbb{R}^d$ is an arbitrary vector for $k\in\mathbb{N}$.

Theorem 3.8 is equivalent to Proposition 3.9 and we refer to the provided proof there. The condition $|\lambda_1|>|\lambda_i|$ for all $i>1$ is satisfied for almost every aggregation matrix $\tilde{\boldsymbol{A}}$. As such, whenever an MPNN of the form in Equation (3.142) is utilized, the collapse of node representations $\boldsymbol{X}^{(k)}$ to a rank-one matrix typically occurs. Our insights also highlight the importance of considering normalized node representations, as the increasing dominance of a single component across all feature channels may otherwise go undetected. Specifically, based on the proof, all columns of these rank-one matrices are scaled versions of the same dominant eigenvector of $\tilde{\boldsymbol{A}}$, induced by the dominant amplification of this eigenvector (CD). We have further related this phenomenon to a special case of power iteration (Kowalewski 1909; Mises et al. 1929) in Section 3.5.4, providing additional understanding. As Theorem 3.8 holds for iteration-dependent feature transformations $\boldsymbol{W}^{(k)}$, it is slightly more general.

3.6.4 Over-Smoothing

Based on Section 3.5.1, we have found that over-smoothing occurs as a special case of rank collapse, when the dominant eigenvector $\boldsymbol{U}_{:,1}$ of $\tilde{\boldsymbol{A}}$ has a particular form. Thus, it can also be seen as the message-passing steps exhibiting SCA and CD, with additional constraints on the specific form of the subspaces. Over-smoothing is typically considered a convergence to a smooth subspace, where either all feature channels become constant vectors or degree-proportional vectors. The dominant eigenvector of $\tilde{\boldsymbol{A}}$ is equal to one of these vectors for established aggregation matrices, e.g., for the symmetrically normalized adjacency matrix $\boldsymbol{A}_{\text{sym}}$, it is $\boldsymbol{D}^{1/2}\mathbf{1}\in\mathbb{R}^n$, and for row-stochastic matrices it is $\mathbf{1}\in\mathbb{R}^n$. As this behavior was related to Laplacian smoothing in the work of Q. Li et al. (2018), it is commonly referred to as over-smoothing. Based on our findings, we define over-smoothing as a special case of rank collapse:

Definition 3.4 (Over-Smoothing). *A sequence of matrices $\left(\boldsymbol{X}^{(k)}\right)_{k\in\mathbb{N}}$ exhibits* ***over-smoothing*** *if it exhibits rank collapse and the sequence of rank-one matrices $(\boldsymbol{Y}^{(k)})_{k\in\mathbb{N}}$ is of the form $\boldsymbol{Y}^{(k)} = \mathbf{1}\left(\boldsymbol{u}^{(k)}\right)^\top$ or $\boldsymbol{Y}^{(k)} = \boldsymbol{D}^{1/2}\mathbf{1}\left(\boldsymbol{u}^{(k)}\right)^\top$ for any sequence of vectors $\left(\boldsymbol{u}^{(k)}\right)_{k\in\mathbb{N}}$.*

For example, in Theorem 3.3, we have provided the novel insight that over-smoothing occurs for the GCN without non-linear activation functions and arbitrary feature transformations:

Theorem 3.9. Let $\boldsymbol{X}^{(k)} = \boldsymbol{A}_{\text{sym}}\boldsymbol{X}^{(k-1)}\boldsymbol{W}^{(k)}$ for a symmetrically normalized adjacency matrix $\boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{n\times n}$ and for a sequence of matrices $\left(\boldsymbol{W}^{(k)}\right)_{k\in\mathbb{N}}$ with $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d\times d}$. Then, for any initial $\boldsymbol{X}^{(0)} \in [-m, m]^{n\times d}$, where $m \in \mathbb{R}$, the sequence $\left(\boldsymbol{X}^{(k)}\right)_{k\in\mathbb{N}}$ exhibits over-smoothing in probability with respect to the Lebesgue measure over $\boldsymbol{X}^{(0)}$.

As Theorem 3.9 is equivalent to Theorem 3.3, we refer to the proof provided there. In Section 3.5.1, we also relate this property to the Dirichlet energy as a commonly employed metric to analyse over-smoothing empirically and theoretically. As $\boldsymbol{L}_{\text{sym}}\boldsymbol{D}^{1/2}\mathbf{1} = \mathbf{0} \in \mathbb{R}^n$, when the Dirichlet energy $E\left(\frac{\boldsymbol{X}^{(k)}}{\|\boldsymbol{X}^{(k)}\|_F}\right)$ converges to zero as $k \to \infty$, the sequence of node representations exhibits over-smoothing. Similarly, we have $\boldsymbol{L}\mathbf{1} = \mathbf{0} \in \mathbb{R}^n$, so when the Dirichlet energy $E_{\boldsymbol{L}}$ computed with the unnormalized graph Laplacian $\boldsymbol{L}$ converges to zero, over-smoothing also occurs. With Proposition 3.10, we have also shown that attention scores with iteration-dependent aggregation matrices $\tilde{\boldsymbol{A}}^{(k)}$ similarly exhibit over-smoothing. Some previous works defined over-smoothing solely as convergence to a constant state. We have shown in Section 3.3.3 that unnormalized node representations $\boldsymbol{X}^{(k)}$ converging to the zero matrix as $k \to \infty$ have been interpreted as convergence to a constant state. In contrast, the normalized node representations converge to a non-constant valued state. In the following, we further summarize our arguments for separating over-smoothing and rank collapse from the norm of the representations.

3.6.5 Vanishing Norm

In Section 3.3.3, we found that the norm of node representations may converge to zero when repeatedly applying message-passing steps, depending on their parameters. In contrast, SCA and CD are properties that describe the relative behavior of different components and their amplification. Consequently, we also defined rank collapse and over-smoothing independently of the norm of the node representations. However, the vanishing norm phenomenon may still significantly affect the performance of MPNNs. Therefore, we argue that the vanishing norm phenomenon warrants further study, albeit separately from SCA, CD, rank collapse, and over-smoothing. By studying these phenomena separately, research efforts can communicate their considered challenge more precisely, develop targeted solutions, and investigate the importance of each fine-grained property more clearly. We formally define vanishing norm as the following property:

Definition 3.5 (Vanishing Norm). *A sequence of matrices $\left(\boldsymbol{X}^{(k)}\right)_{k\in\mathbb{N}}$ exhibits **vanishing norm** if*

$$\lim_{k\to\infty}\left\|\boldsymbol{X}^{(k)}\right\|_F = 0\,. \tag{3.148}$$

Based on Proposition 3.3, we identified vanishing norm to occur in MPNNs when the learnable parameters across all iterations are sufficiently close to zero. The vanishing norm property is closely related to vanishing gradients, which have been a key challenge in MLPs and RNNs (Bengio et al. 1994; Hochreiter et al. 1997; Pascanu et al. 2013).

3.6.6 Quantifying Rank Collapse and Over-Smoothing

Based on the insights and definitions we have provided, we now introduce a novel metric for quantifying rank collapse, which generalizes over-smoothing. This will enable us to empirically analyze the effectiveness of further methods without the need to study them theoretically. When using metrics to quantify over-smoothing, many instances of the more general phenomenon of rank collapse would remain undetected. Unifying them into a common metric to identify all methods leading to rank collapse, including all forms of over-smoothing, would be ideal. Based on our theoretical insights and Definition 3.3 on rank collapse, we propose to quantify the similarity of a sequence of states to rank-one matrices. The left and right singular vectors of the maximal singular value give the best rank-one approximation. While we recommend computing this distance, we also propose a more efficient method when obtaining the left and right singular vectors is not computationally feasible. When $\boldsymbol{X}$ is a rank-one matrix, any row and any column allow us to determine all values of $\boldsymbol{X}$. Consequently, choosing any row and column of $\boldsymbol{X}$ and forming a normalized rank-one matrix using their outer product and computing the distance to $\frac{\boldsymbol{X}}{\|\boldsymbol{X}\|_2}$ approximates the similarity to a rank-one matrix. We refer to this metric as the rank-one distance (ROD) and define it as follows:

Definition 3.6 (Rank-One Distance (ROD)). *Let $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ be a matrix, $\boldsymbol{u} = \boldsymbol{X}_{:,i}$ a column of $\boldsymbol{X}$ with $i = \arg\max_k \|\boldsymbol{X}_{:,k}\|$ and $j = \arg\max_k \|\boldsymbol{X}_{k,:}\|$ the row of $\boldsymbol{X}$ with maximal norm. To account for the correct sign, we set $\boldsymbol{v} = \boldsymbol{X}_{j,:}$ if $X_{j,i} > 0$ and $\boldsymbol{v} = -\boldsymbol{X}_{j,:}$ otherwise. The rank-one distance (ROD) is defined as*

$$\mathit{ROD}(\boldsymbol{X}) = \left\|\frac{\boldsymbol{X}}{\|\boldsymbol{X}\|_2} - \frac{\boldsymbol{u}\boldsymbol{v}^\top}{\|\boldsymbol{u}\boldsymbol{v}^\top\|_2}\right\|_*, \tag{3.149}$$

where $\|\cdot\|_*$ is the nuclear norm.

Our definition of ROD aligns well with Definition 3.3 for rank collapse. With this metric, we can identify methods that exhibit rank collapse without requiring a theoretical analysis. Similar to over-smoothing being a particular case of rank collapse, the Dirichlet energy of the normalized state converging to zero is a special case of the ROD converging to zero.

3.6.7 Avoiding Rank Collapse and Over-Smoothing

Based on our findings, rank collapse can be prevented by dealing with either SCA or CD. To recall the notation, we restate the vectorized form of our considered message-passing iterations given by Equation (3.142) as

$$\operatorname{vec}\left(\boldsymbol{X}^{(k)}\right) = \boldsymbol{T}^{(k)} \operatorname{vec}\left(\boldsymbol{X}^{(k-1)}\right) \tag{3.150}$$

where $\boldsymbol{T}^{(k)} = \left(\boldsymbol{W}^{(k)}\right)^\top \otimes \tilde{\boldsymbol{A}} \in \mathbb{R}^{nd \times nd}$ for some $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d \times d}$ and $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n \times n}$. A central aspect of SCA in these MPNNs is that all eigenvectors and singular vectors of $\boldsymbol{T}^{(k)}$ are given as Kronecker products of eigenvectors and singular vectors of $\boldsymbol{W}^{(k)}$ and $\tilde{\boldsymbol{A}}$. This property always holds when $\boldsymbol{T}^{(k)}$ can be decomposed as a Kronecker product. To avoid SCA, it is necessary to use a transformation $\boldsymbol{T}^{(k)}$ that is not decomposable into a single Kronecker product. However, applying a dense matrix $\boldsymbol{T}^{(k)}$ of shape $(nd \times nd)$ is computationally infeasible. We aim to remain within the framework of applying feature transformations and aggregation matrices. This can be achieved by noting that any matrix

$$\boldsymbol{T}^{(k)} = \sum_{i=1}^{r} \boldsymbol{W}_{(i)}^{(k)} \otimes \tilde{\boldsymbol{A}}_{(i)} \tag{3.151}$$

can be decomposed into a sum of Kronecker products (SKP) (Cao et al. 2021) for some $\boldsymbol{W}_{(i)}^{(k)} \in \mathbb{R}^{d \times d}$ and $\tilde{\boldsymbol{A}}_{(i)} \in \mathbb{R}^{n \times n}$ for all $i \in [r]$. For any matrix $\boldsymbol{T}^{(k)}$, there exists a trivial decomposition into a sum of $r = d^2$ Kronecker products, where each $\boldsymbol{W}_{(i)}$ has a single non-zero entry and $\tilde{\boldsymbol{A}}_{(i)}$ contains the corresponding part of $\boldsymbol{T}^{(k)}$. Applying this choice to the node representations as in Equation (3.150), i.e., $\boldsymbol{T}^{(k)} \operatorname{vec}\left(\boldsymbol{X}^{(k-1)}\right)$, in the usual matrix form results in the iterative form

$$\boldsymbol{X}^{(k)} = \sum_{i=1}^{r} \tilde{\boldsymbol{A}}_{(i)} \boldsymbol{X}^{(k-1)} \boldsymbol{W}_{(i)}^{(k)} . \tag{3.152}$$

This SKP-based formulation allows us to avoid storing or applying $\boldsymbol{T}^{(k)} \in \mathbb{R}^{nd \times nd}$ directly. Instead, the matrices $\tilde{\boldsymbol{A}}_{(1)}, \ldots, \tilde{\boldsymbol{A}}_{(r)}$ and $\boldsymbol{W}_{(1)}, \ldots, \boldsymbol{W}_{(r)}$ are stored and applied separately. This formulation is capable of maximally amplifying distinct components in the node representations across different feature channels, avoiding SCA. When each aggregation matrix $\tilde{\boldsymbol{A}}_{(i)}$ has different eigenvectors and singular vectors, the corresponding feature transformation $\boldsymbol{W}_{(i)}^{(k)}$ controls the mixing of the amplified components for each feature channel. As such, for r pairs of aggregation and feature transformation matrices, we can amplify at least r different components across r different feature channels. We formalize this in the following proposition:

Proposition 3.12 (An SKP can avoid SCA). Let $\boldsymbol{V} \in \mathbb{R}^{n \times d}$ with $\operatorname{rank}(\boldsymbol{V}) = r$. Then, there exist matrices $\tilde{\boldsymbol{A}}_{(1)}, \ldots, \tilde{\boldsymbol{A}}_{(r)} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{W}_{(1)}, \ldots, \boldsymbol{W}_{(r)} \in \mathbb{R}^{d \times d}$ such that for any matrix $\boldsymbol{P} \in \mathbb{R}^{n \times d}$ with $\boldsymbol{P} \neq p \cdot \boldsymbol{V}$ for any $p \in \mathbb{R}$, and $\|\boldsymbol{P}\|_F = \|\boldsymbol{V}\|_F$, the following inequality holds:

$$\left\| \sum_{i=1}^{r} \tilde{\boldsymbol{A}}_{(i)} \boldsymbol{V} \boldsymbol{W}_{(i)} \right\|_F > \left\| \sum_{i=1}^{r} \tilde{\boldsymbol{A}}_{(i)} \boldsymbol{P} \boldsymbol{W}_{(i)} \right\|_F . \tag{3.153}$$

Proof. Without loss of generality, assume that the first r columns of $\boldsymbol{V}$ are linearly independent. Define the matrices $\tilde{\boldsymbol{A}}_{(i)} = \frac{\boldsymbol{V}_{:,i}\boldsymbol{V}_{:,i}^\top}{\left\|\boldsymbol{V}_{:,i}\boldsymbol{V}_{:,i}^\top\right\|_F}$, which are symmetric and rank-one. Let $\boldsymbol{W}_{(i)}$ be the matrix with a one at position (i, i) and zeros at all other positions, i.e., $\left(\boldsymbol{W}_{(i)}\right)_{i,i} = 1$. The matrix $\boldsymbol{S} = \sum_{i=1}^{r} \left(\boldsymbol{W}_{(i)}\right)^\top \otimes \tilde{\boldsymbol{A}}_{(i)}$ has an eigenvector $\mathrm{vec}(\boldsymbol{V})$ with corresponding eigenvalue $\lambda_1 = 1$, and all other eigenvalues of $\boldsymbol{S}$ are zero. Using the equivalent vector norm, we thus have $\|\boldsymbol{S}\,\mathrm{vec}(\boldsymbol{V})\|_2 = \lambda_1\|\boldsymbol{V}\|_F$ and $\|\boldsymbol{S}\,\mathrm{vec}(\boldsymbol{P})\|_2 < \lambda_1\|\boldsymbol{P}\|_F$ for all vectors $\mathrm{vec}(\boldsymbol{P}) \neq p\,\mathrm{vec}(\boldsymbol{V})$ for any $p \in \mathbb{R}$. The assumption $\|\boldsymbol{V}\|_F = \|\boldsymbol{P}\|_F$ concludes the proof. □

Proposition 3.12 confirms that a sum of r Kronecker products can maximally amplify any desired features of rank r. This finding confirms that multiple terms effectively prevent SCA and, consequently, rank collapse and over-smoothing. Message-passing methods utilizing a single edge weight between any two nodes can also be seen as using a single computational graph $\tilde{\boldsymbol{A}}$. In contrast, our proposed SKP corresponds to the use of multiple computational graphs $\tilde{\boldsymbol{A}}_{(1)}, \ldots, \tilde{\boldsymbol{A}}_{(r)}$. While each $\tilde{\boldsymbol{A}}_{(i)}$ can be interpreted as a distinct graph with independent edges and weights, they can also represent multiple edge relation types over a shared node set. Such a combination of $\tilde{\boldsymbol{A}}_{(1)}, \ldots, \tilde{\boldsymbol{A}}_{(r)}$ is also referred to as a multigraph. While each computational graph may utilize a distinct structure, a shared structure with different edge weights already suffices for amplifying distinct components.

With this formulation, each aggregation matrix $\tilde{\boldsymbol{A}}_{(i)}$ can amplify distinct components, and the learnable parameters $\boldsymbol{W}_{(i)}$ control how strongly these components are amplified for each feature channel. This enables more effective optimization, as the parameters have a greater impact on the resulting representations. Even with $r = 2$ computational graphs, an SKP can amplify distinct components across feature channels, thereby mitigating SCA and rank collapse.

This finding provides a general framework for MPNNs that utilize an SKP, or equivalently, multiple computational graphs. This SKP-based framework is not tied to specific choices of aggregation functions, but rather establishes a general desirable property for MPNNs when the goal is to amplify distinct components across feature channels. Many choices for aggregation functions and feature transformations are possible within the SKP framework. The SKP form is similar to multi-head attention, where each $\tilde{\boldsymbol{A}}_{(i)}$ corresponds to an attention head. However, as these are softmax-activated, each $\tilde{\boldsymbol{A}}_{(i)}$ is a row-stochastic matrix, which all have the all-ones vector $\mathbf{1}$ as their dominant eigenvector. Our proof requires distinct dominant eigenvectors so that each parameter matrix $\boldsymbol{W}_{(i)}$ controls which components are maximally amplified in each feature channel.

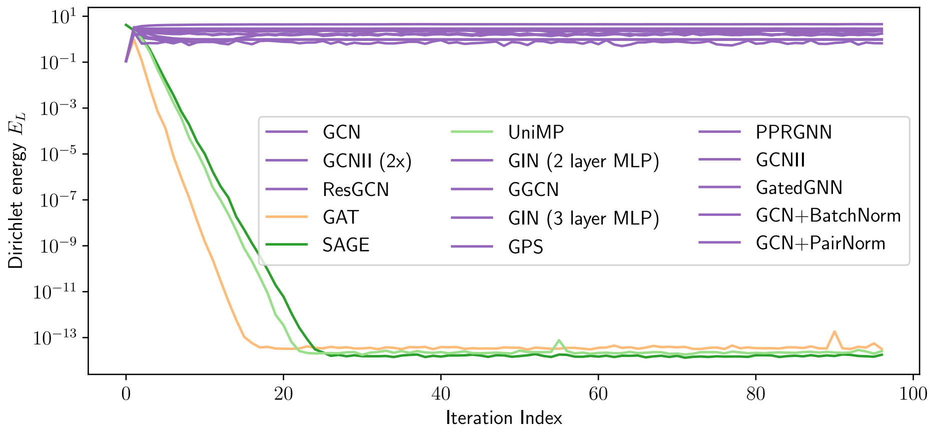


Figure 3.12: Dirichlet energy $E_{\boldsymbol{L}}\left(\boldsymbol{X}^{(k)}\right) = \operatorname{tr}\left(\left(\boldsymbol{X}^{(k)}\right)^{\top}\boldsymbol{L}\boldsymbol{X}^{(k)}\right)$ using the unnormalized graph Laplacian $\boldsymbol{L}$ across 96 iterations of message passing for the KarateClub dataset (Zachary 1977) and 15 methods. All parameters are randomly initialized, and visualized values are averaged over 50 runs.

This finding also reveals a connection between convolutional neural networks (CNNs) (Fukushima 1980; LeCun, Boser, et al. 1989) and the avoidance of SCA in message-passing. Convolution on a grid can be interpreted as a special case of an SKP with r terms, where r corresponds to the number of elements in the filter or kernel. Each relative connection in the convolution can be represented by a connectivity matrix $\tilde{\boldsymbol{A}}_{(i)}$ over positions on the grid, with a corresponding distinct linear transformation $\boldsymbol{W}_{(i)}$ applied for each position. This perspective helps explain why SCA — and, consequently, over-smoothing — is a phenomenon that arises specifically in the context of graph neural networks.

3.7 Evaluation

We now empirically validate our theoretical insights and extend them to additional MPNNs and non-linear activation functions. Based on the definition of rank collapse in Definition 3.6 and the corresponding rank-one distance (ROD) metric, we first evaluate several existing methods in their ability to avoid rank collapse and compare this metric to two versions of the Dirichlet energy. As the second part, we evaluate a sum of Kronecker products (SKP) within MPNNs, rather than a single Kronecker product, to evaluate its ability to approximate target functions for various benchmark tasks.

3.7.1 Quantifying Rank Collapse Using ROD

Based on our theoretical insights and definitions for rank collapse and over-smoothing, we introduced the rank-one distance (ROD) as a metric capturing a more general phenomenon than over-smoothing. Here, we compare the Dirichlet energy as a metric to

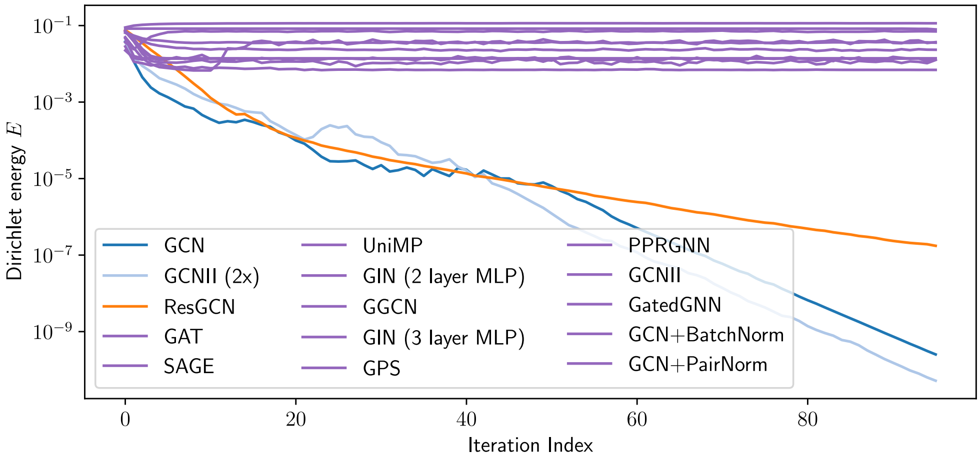


Figure 3.13: Dirichlet energy $E\left(\boldsymbol{X}^{(k)}\right) = \operatorname{tr}\left(\left(\boldsymbol{X}^{(k)}\right)^{\top} \boldsymbol{L}_{\text{sym}} \boldsymbol{X}^{(k)}\right)$ using the symmetrically normalized graph Laplacian $\boldsymbol{L}_{\text{sym}}$ across 96 iterations of message passing for the KarateClub dataset (Zachary 1977) and 15 methods. All parameters are randomly initialized, and visualized values are averaged over 50 runs.

quantify over-smoothing with the ROD. Instead of only indicating whether the node representations are similar to a rank-one matrix formed by specific columns, ROD determines the similarity to any rank-one matrix. We evaluate these metrics for several established methods, allowing us to gain insights into the ability of those methods to avoid rank collapse or over-smoothing. We provide a reproducible implementation of these experiments [1]. These experiments are based on (Roth 2024).

Experimental Setup We consider the update functions of various MPNNs, interleaved with a nonlinear activation function. In general, these are of the form

$$\boldsymbol{X}^{(k)} = \phi\left(f\left(\boldsymbol{A}, \boldsymbol{X}^{(k-1)}\right)\right) \tag{3.154}$$

where ϕ is set to be the ReLU activation function, and f is a message-passing method, which we will detail in the following. For the graph structure $\boldsymbol{A}$, we use the small KarateClub dataset (Zachary 1977). It consists of a single undirected graph with $n = 34$ nodes and 156 edges. Initial node features $\boldsymbol{X}^{(0)} \in \mathbb{R}^{n \times d}$ are randomly initialized from a normal distribution with $X_{i,j} \sim \mathcal{N}(0,1)$. We consider three metrics based on the normalized node representations $\frac{\boldsymbol{X}^{(k)}}{\left\|\boldsymbol{X}^{(k)}\right\|_F}$: The Dirichlet energy

$$E_{\boldsymbol{L}}\left(\frac{\boldsymbol{X}^{(k)}}{\left\|\boldsymbol{X}^{(k)}\right\|_F}\right) = \frac{\operatorname{tr}\left(\left(\boldsymbol{X}^{(k)}\right)^{\top} \boldsymbol{L} \boldsymbol{X}^{(k)}\right)}{\left\|\boldsymbol{X}^{(k)}\right\|_F^2} \tag{3.155}$$

[1] https://github.com/roth-andreas/simplifying-over-smoothing

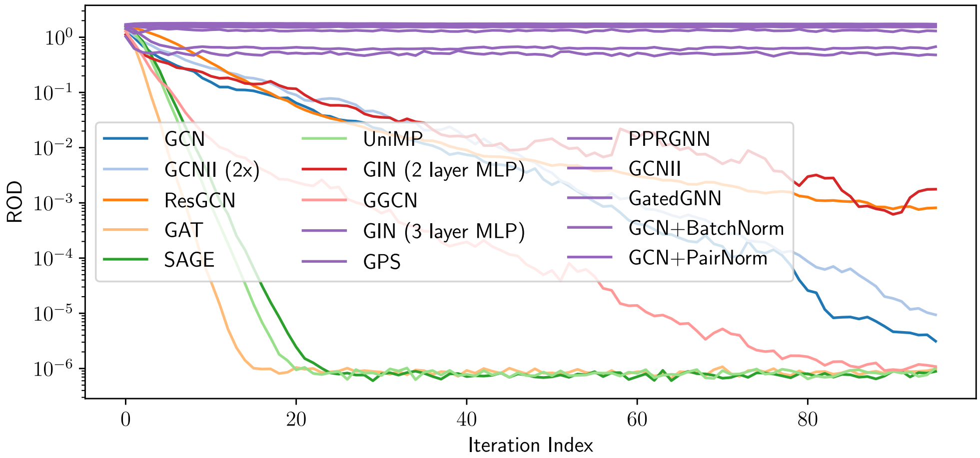


Figure 3.14: Rank-one distance (ROD) across 96 iterations for the KarateClub dataset (Zachary 1977) and 15 methods. All parameters are randomly initialized, and visualized values are averaged over 50 runs.

utilizing the unnormalized graph Laplacian $\boldsymbol{L} \in \mathbb{R}^{n\times n}$, which is zero when $\boldsymbol{X}^{(k)} = \mathbf{1}\boldsymbol{u}^\top$ for any $\boldsymbol{u} \in \mathbb{R}^d$. As the second metric, we evaluate the Dirichlet energy

$$E\left(\frac{\boldsymbol{X}^{(k)}}{\left\|\boldsymbol{X}^{(k)}\right\|_F}\right) = \frac{\operatorname{tr}\left(\left(\boldsymbol{X}^{(k)}\right)^\top \boldsymbol{L}_{\text{sym}}\boldsymbol{X}^{(k)}\right)}{\left\|\boldsymbol{X}^{(k)}\right\|_F^2} \tag{3.156}$$

utilizing the symmetrically normalized graph Laplacian $\boldsymbol{L}_{\text{sym}}$, which is zero when $\boldsymbol{X}^{(k)} = \boldsymbol{D}^{1/2}\mathbf{1}\boldsymbol{u}^\top$ for any $\boldsymbol{u} \in \mathbb{R}^d$. Lastly, the rank-one distance (ROD), which is zero for any rank-one matrix $\boldsymbol{X}^{(k)} = \boldsymbol{v}\boldsymbol{u}^\top$ for any $\boldsymbol{v} \in \mathbb{R}^n$ and $\boldsymbol{u} \in \mathbb{R}^d$, as defined in Definition 3.6. We compute all metrics for $\boldsymbol{X}^{(k)}$ for each $k \in [0, 96]$. With more iterations, numerical instabilities prevented us from computing meaningful values.

Evaluated Models We consider several established MPNNs as function f in Equation (3.154). The first set of methods utilizes the symmetrically normalized adjacency matrix $\boldsymbol{A}_{\text{sym}}$ for aggregation. The dominant eigenvector of $\boldsymbol{A}_{\text{sym}}$ is the degree-proportional vector. These are the graph convolutional network (GCN) (Kipf et al. 2017) and the GCN with the previous state added to the output using a residual connection (K. He et al. 2016) as in the ResGCN (Kipf et al. 2017). We also evaluate GCNII (M. Chen et al. 2020) that further adds initial residual connections that combine the initial state with the output of each iteration. As we observed differences depending on the magnitude of feature transformation, we further evaluate a variant in which all feature transformations are scaled by a factor of 2 (GCNII ($2\times$)). Additionally, we consider the personalized PageRank graph neural network (PPRGNN) (Roth and Liebig 2022b), which proposes a weighting scheme on the initial residual connections that ensures convergence of node representations to a fixed point for an infinite number of iterations while maintaining a portion of the initial representations.

The second set of methods utilizes an averaging function as the aggregation function, potentially weighted using a learnable function. These are the graph attention network (GAT) (Velickovic et al. 2018), and the SAGE convolution (Hamilton et al. 2017). The dominant eigenvector of matrices representing such an averaging aggregation is the constant vector. As graph transformers similarly average over all nodes in a graph, we evaluate the unified message-passing model (UniMP) (Shi et al. 2021) and the general, powerful, and scalable graph transformer (GPS) (Rampásek, Galkin, et al. 2022), which combines the graph transformer framework with local message-passing and feature normalization.

We also evaluate several other methods that were specifically proposed to mitigate over-smoothing, regarding their ability to avoid the more general rank collapse. The generalized GCN (GGCN) (Yan et al. 2022) allows for negative edge weights, which they have shown to reduce the smoothing effect between adjacent nodes.

Additionally, we consider adding normalization layers. We evaluate batch normalization (Ioffe et al. 2015) as a well-established normalization procedure and PairNorm (Zhao et al. 2020) as a normalization that was specifically proposed to prevent over-smoothing. We combine both normalization techniques with the GCN to evaluate their ability to prevent rank collapse for a method known to suffer from over-smoothing.

As another established mechanism within MPNNs, we evaluate the ability of gating to prevent rank collapse. These use gating coefficients to control the mixing of aggregated features with the previous state. Here, we evaluate the gated graph neural network (GatedGNN) (Y. Li et al. 2016).

While the above methods apply linear feature transformation, the graph isomorphism network (GIN) (Xu, Hu, et al. 2019) employs a multi-layer perceptron (MLP), which is known to be a universal approximator (Hornik et al. 1989). We evaluate two such instantiations: one that applies a two-layer MLP (GIN 2-layer MLP), and one that uses a three-layer MLP (GIN 3-layer MLP) as a feature transformation.

We use the standard implementation from Pytorch-Geometric (Fey and Lenssen 2019) when available, or use the implementation provided with the proposed methods. All learnable parameters are randomly initialized based on their provided implementations. We repeat this process for 50 random seeds.

Results We present mean values for the Dirichlet energy $E_{\boldsymbol{L}}\left(\frac{\boldsymbol{X}^{(k)}}{\|\boldsymbol{X}^{(k)}\|_F}\right)$ using the unnormalized graph Laplacian in Figure 3.12. We observe that this Dirichlet energy quickly decreases to small values for GAT, SAGE, and UniMP. After around 20 iterations, the Dirichlet energy of these models stabilizes at around 10^{-13}, which appears to be the machine precision. These three models utilize a (weighted) mean aggregation, which is known to have the constant vector as the dominant eigenvector. All other methods remain roughly constant throughout all iterations.

For comparison, we also present mean values for the Dirichlet energy $E\left(\frac{\boldsymbol{X}^{(k)}}{\|\boldsymbol{X}^{(k)}\|_F}\right)$ using the symmetrically normalized graph Laplacian in Figure 3.13. We observe this Dirichlet energy to decrease towards zero with an increased number of iterations for the GCN, ResGCN, and GCNII ($2\times$). These methods utilize the symmetrically normalized adjacency matrix as their aggregation function. While convergence to zero can be proven for GCN and ResGCN in the linear case, for GCNII ($2\times$), the influence of the constant initial features seemingly diminishes as the number of iterations increases. All other methods remain at a roughly constant Dirichlet energy.

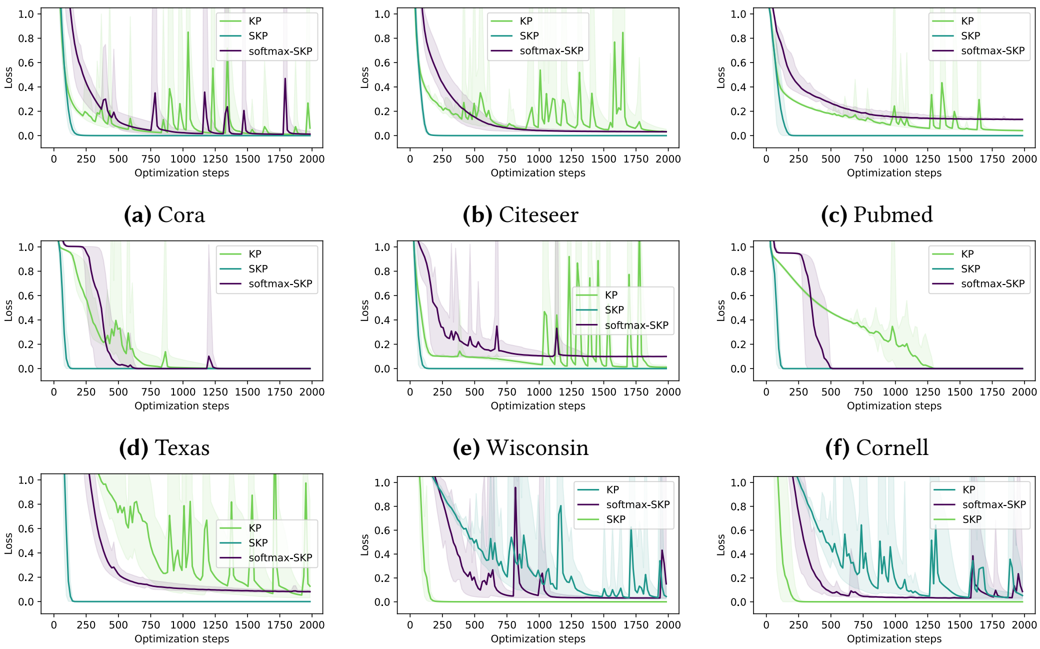
As the third metric, we evaluate the ROD in Figure 3.14. The ROD decreases to zero when $\boldsymbol{X}^{(k)}$ becomes closer to any rank-one matrix. We observe this to be the case for all methods, for which either one of $E_{\boldsymbol{L}}\left(\frac{\boldsymbol{X}^{(k)}}{\left\|\boldsymbol{X}^{(k)}\right\|_F}\right)$ or $E\left(\frac{\boldsymbol{X}^{(k)}}{\left\|\boldsymbol{X}^{(k)}\right\|_F}\right)$ decreases to zero, as these are special cases. Additionally, we observe the ROD for the GIN (2-layer MLP) to approach zero. Since the sum aggregation has neither a constant nor a degree-proportional vector as its dominant eigenvector, the representations appear to converge to a different vector that is shared across all feature channels. The GIN utilizing a 3-layer MLP as a feature transformation successfully avoids this rank-one state. Additionally, we observe that the node representations for the GGCN, which allows for negative edge weights, also converge to a rank-one matrix. While negative edge weight can cause the dominant eigenvector to also be significantly different, GGCN still utilizes a single computational graph, exhibits SCA, and amplifies the same signal component across all feature channels.

Methods that avoid rank collapse include both normalization methods, GCN+BatchNorm and GCN+PairNorm, as well as the methods GatedGNN, PPRGNN, GCNII, GPS, and GIN (3-layer MLP). These appear to be promising directions for addressing the underlying problem of rank collapse.

3.7.2 The Effectiveness of a Sum of Kronecker Products

As our theoretical findings indicate a limited effectiveness of learnable parameters for common forms of message passing, we now empirically investigate the optimization properties of several message-passing frameworks. For this, we consider nine common benchmark datasets for node classification and track the ability of different message-passing frameworks to fit the training data. Ideally, employed methods are sufficiently expressive in their ability to closely approximate desired functions. We provide a reproducible implementation of these experiments [2]. This set of experiments is based on Roth and Liebig (2023).

[2] https://github.com/roth-andreas/rank_collapse



(a) Cora **(b)** Citeseer **(c)** Pubmed

(d) Texas **(e)** Wisconsin **(f)** Cornell

(g) Film **(h)** Chameleon **(i)** Squirrel

Figure 3.15: Training error during optimization for nine benchmark tasks for node classification when using all labels for optimization. The evaluated methods are a Kronecker product (KP), a sum of Kronecker products (SKP), and a sum of Kronecker products where each aggregation function is node-wise softmax-activated (softmax-SKP). All edge weights and feature transformations are directly learnable. Each method uses $l = 8$ iterations and $d = 32$ feature channels. Average values across 10 runs are in bold, with the minimum and maximum values presented as a shaded area.

Datasets We consider nine graphs for node classification. These are Cora, Citeseer, and Pubmed (Z. Yang et al. 2016), which are citation networks for which each node corresponds to a document, and a reference between documents is represented as an edge. The task is to classify each document based on a bag-of-words representation and the references. These are considered to be homophilic datasets, as referenced documents mostly share the same class (Yan et al. 2022). The other six considered datasets are referred to as heterophilic datasets, as these contain fewer shared classes between neighboring nodes (Yan et al. 2022). These consist of three web page graphs: Texas, Wisconsin, Cornell (Pei et al. 2020), each of which represents web pages from one university and their links, as nodes and edges, respectively. The task is to classify each web page into one of the categories student, project, course, staff, and faculty (Pei et al. 2020). The next two datasets are the Wikipedia graphs Chameleon (Rozemberczki et al. 2021) and Squirrel (Rozemberczki et al. 2021). Similarly, nodes correspond to web pages from Wikipedia related to the corresponding topic, and edges correspond to their mutual links. The task is to categorize each web page based on its monthly traffic. Lastly, the actor co-occurrence dataset Film (Tang et al. 2009) represents actors as nodes and their co-occurrence on

Wikipedia web pages as edges. The task is to categorize the number of words on the Wikipedia page of the actor. For each dataset, we utilize the largest connected component. For this experiment, we aim to approximate the target function and assume all node labels are available for optimization using the cross-entropy loss.

Experimental Setup Given one of these graphs with n nodes and c labels, we use the provided initial node representations $\boldsymbol{X} \in \mathbb{R}^{n \times h}$, where h is the input feature dimension of the corresponding task, and discrete node labels $\boldsymbol{Y} \in \{0,1\}^{n \times c}$ encoded as a one-hot vector for each node. We first apply a linear feature transformation to obtain node representations $\boldsymbol{X}^{(0)} \in \mathbb{R}^{n \times d}$ with the number of feature channels $d \in \mathbb{N}$ being a tunable parameter. We then update the node representations

$$\operatorname{vec}\left(\boldsymbol{X}^{(k)}\right) = \phi\left(\boldsymbol{T}^{(k)} \operatorname{vec}\left(\boldsymbol{X}^{(k-1)}\right)\right) \tag{3.157}$$

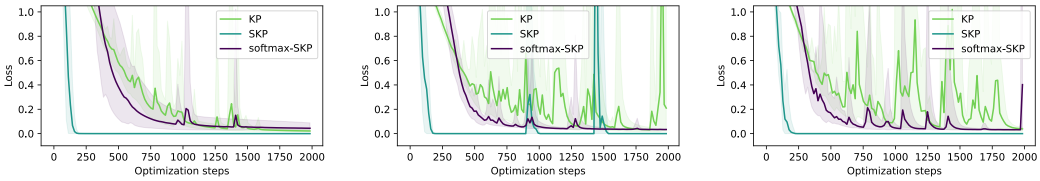
using an iteration-dependent linear message-passing step $\boldsymbol{T}^{(k)}$ and extend our theoretical insights by incorporating a non-linear activation function ϕ. We introduce our different choices for $\boldsymbol{T}^{(k)}$ in the following paragraph. After l such iterations, each node representation $\boldsymbol{X}_{i,:}^{(l)}$ is mapped to class predictions $\hat{\boldsymbol{Y}} \in [0,1]^{n \times c}$ using an affine transformation and a softmax activation. The cross-entropy loss between $\boldsymbol{Y}$ and $\hat{\boldsymbol{Y}}$ is computed and optimized for 2000 steps using the Adam optimizer (Kingma et al. 2015) and a learning rate of 0.001. For each framework chosen for $\boldsymbol{T}^{(k)}$, this process is repeated for ten random initializations.

Methods Based on our theoretical findings, we evaluate three different frameworks as the message-passing function $\boldsymbol{T}^{(k)}$ in Equation (3.157): a single Kronecker product, a sum of Kronecker products, and a sum of Kronecker products for which each aggregation matrix is softmax activated. We extend these methods slightly from our theory to provide further insights by allowing for iteration-dependent aggregation matrices and a non-linear activation function after each iteration.

The single Kronecker product corresponds to the established framework that utilizes a single feature transformation $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d \times d}$ and a single aggregation matrix $\tilde{\boldsymbol{A}}^{(k)} \in \mathbb{R}^{n \times n}$ for each iteration k, i.e., we have

$$\boldsymbol{T}_{\text{KP}}^{(k)} = \boldsymbol{W}^{(k)} \otimes \tilde{\boldsymbol{A}}^{(k)} \,. \tag{3.158}$$

Most of our theoretical insights have assumed that the message-passing is of this form, as it is used within many established approaches, such as GCN (Kipf et al. 2017) and GraphSAGE (Hamilton et al. 2017). It can also represent the sum aggregation (Xu, Hu, et al. 2019), negative edge weights (Bo et al. 2021), and many more (Chien et al. 2021; Yan et al. 2022). To cover all of these approaches in our experiments, we allow for all entries of the feature transformation $\boldsymbol{W}^{(k)}$ and the weights of all given edges to be directly learnable for $\tilde{\boldsymbol{A}}^{(k)}$ for each $k \in \mathbb{N}$. Based on our theoretical insights, a single Kronecker product



(a) $d = 16$ **(b)** $d = 64$ **(c)** $d = 128$

Figure 3.16: Training error during optimization for the squirrel benchmark tasks when using all labels for optimization, as in Figure 3.15. The number of feature channels d is varied in $d \in \{16, 64, 128\}$ and the number of iterations is set as $l = 8$. Average values across 10 runs are in bold, with the minimum and maximum values presented as a shaded area.

exhibits shared component amplification, limiting the effectiveness of such approaches (see Section 3.6).

Next, we evaluate the effectiveness of our identified solution to shared component amplification using a sum of Kronecker products (see Section 3.6.7). For this, we utilize $h = 2$ terms and set

$$\boldsymbol{T}_{\text{SKP}}^{(k)} = \sum_{i=1}^{2} \boldsymbol{W}_{(i)}^{(k)} \otimes \tilde{\boldsymbol{A}}_{(i)}^{(k)} \tag{3.159}$$

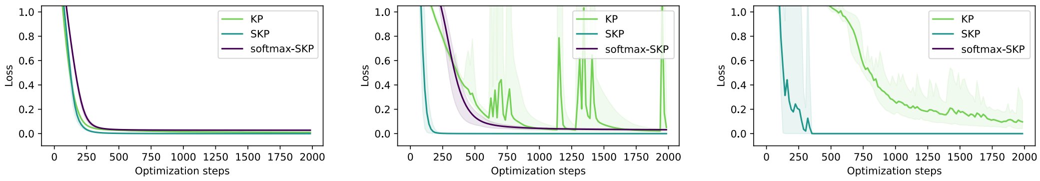
where $\boldsymbol{W}_{(i)}^{(k)} \in \mathbb{R}^{d \times d}$ and $\tilde{\boldsymbol{A}}_{(i)}^{(k)} \in \mathbb{R}^{n \times n}$ for $i \in \{1, 2\}$. As with $\boldsymbol{T}_{\text{KP}}^{(k)}$, the entries of $\tilde{\boldsymbol{A}}_{(1)}^{(k)}$ and $\tilde{\boldsymbol{A}}_{(2)}^{(k)}$ corresponding to existing edges are directly learnable. Based on Proposition 3.12, this form enables more effective feature transformations, as it allows for controlled mixing of amplified components across different feature channels.

The third considered framework is inspired by attention-based methods, such as the graph attention network (GAT) (Velickovic et al. 2018), the TransformerConv (Shi et al. 2021) (Brody et al. 2022). The message-passing step is defined as

$$\boldsymbol{T}_{\text{softmax}-\text{SKP}}^{(k)} = \sum_{i=1}^{2} \boldsymbol{W}_{(i)}^{(k)} \otimes \tau\left(\tilde{\boldsymbol{A}}_{(i)}^{(k)}\right) \tag{3.160}$$

where τ applies a row-wise softmax activation function, as is used for attention-based aggregation functions. As with $\boldsymbol{T}_{\text{KP}}^{(k)}$ and $\boldsymbol{T}_{\text{SKP}}^{(k)}$, all edge weights in $\tilde{\boldsymbol{A}}_{(1)}^{(k)}$ and $\tilde{\boldsymbol{A}}_{(2)}^{(k)}$ are directly learnable.

Results In Figure 3.15, we visualize the training errors during optimization for all nine datasets, using $l = 8$ message-passing iterations and $d = 32$ feature channels. We observe that the SKP consistently achieves smooth error minimization, resulting in an error close to zero after fewer than 250 optimization steps for all datasets. For KP and softmax-SKP, the error minimization is slower in all cases and exhibits unstable behavior. For many datasets, the error after 2000 optimization steps also remains higher than the obtained



(a) $l = 1$ **(b)** $l = 4$ **(c)** $l = 32$

Figure 3.17: Training error during optimization for the squirrel benchmark tasks when using all labels for optimization, as in Figure 3.15. The number of iterations l is varied in $l \in \{1, 4, 32\}$ and the number of feature channels is set as $d = 32$. Average values across 10 runs are in bold, with the minimum and maximum values presented as a shaded area.

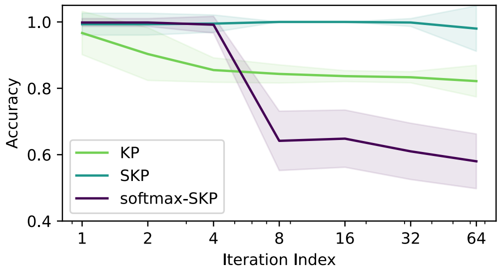


Figure 3.18: Obtained training accuracy for methods with a number of iterations $l \in \{1, 2, 4, 8, 16, 32, 64\}$ for the synthetic 3-class classification task with four nodes. Average accuracy over 10 runs in bold, standard deviation as shaded area.

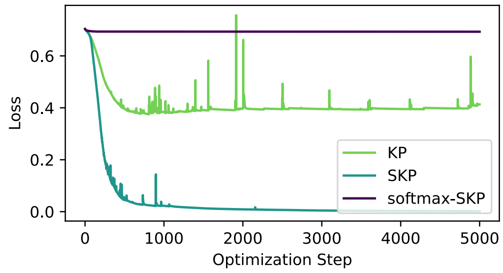


Figure 3.19: Optimization error for $l = 8$ iterations and the synthetic 3-class classification task with four nodes. Averaged over 50 random seeds.

loss for SKP. The smoothness of the optimization and the convergence speed both indicate that optimizing the parameters within the SKP framework is more effective than for the KP framework or the softmax-SKP framework.

We additionally present an ablation study varying the number of feature channels $d \in \{16, 64, 128\}$ for the Squirrel dataset in Figure 3.16. While the optimization remains fast and smooth for the SKP in all cases, it remains slower and unstable for both the KP and softmax-SKP. Even when the SKP uses $d = 16$ feature channels, the optimization is significantly better than the KP and softmax-SKP, even with $d = 256$ feature channels. This aligns with our theoretical findings, as we identified that the KP amplifies components shared across all feature channels, making additional feature channels redundant.

In Figure 3.17, we present an additional ablation study that varies the number of iterations. For a single iteration ($l = 1$), the optimization remains smooth for all methods, but the SKP minimizes the error slightly faster and achieves a slightly lower error. However, the difference is rather small. Aligning with our theoretical insights, the more iterations l

we employ, the more significant the difference becomes. While for $l = 32$ iterations, SKP remains mostly smooth and achieves an error close to zero, the error for KP and softmax-SKP is increasingly unstable, and convergence is slower.

Synthetic Dataset

We additionally extend this experiment to a more challenging synthetic task. When node representations collapse into a lower-dimensional subspace, it becomes more challenging to assign them to different classes. We begin with an observation about three nodes of a graph. When their node representations collapse into a one-dimensional subspace, the representations are positioned at a specific point on a line. One node is located in the center, with each of the two other nodes positioned on opposite sides. A linear classifier cannot classify the two outer nodes in the same class while classifying the center node in a different class. Representations need to be in a higher-dimensional subspace for solving this task. Our synthetic task follows this observation. We consider a slightly more challenging version with four nodes and the multi-label classification task, where the labels are given as

$$\mathbf{Y} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.161}$$

Each pair of two nodes shares the same label for one of the tasks. When node representations lie in a subspace of two or fewer dimensions, a linear decision boundary cannot correctly classify all nodes. Thus, this task is challenging for methods that yield node representations in a lower-dimensional subspace.

Experimental Setup Initial node representations $\mathbf{X}^{(0)} \in \mathbb{R}^{4\times 6}$ are sampled from a normal distribution with mean zero and standard deviation set to one. Aggregation matrices and feature transformation are randomly initialized from a normal distribution. As we consider a multi-label classification task, the predictions are obtained using a sigmoid activation, rather than the softmax activation used for node classification tasks. We consider methods with $l \in \{1, 2, 4, 8, 16, 32, 64\}$ iterations for message-passing. Optimization for each method and number of iterations is separately performed for 5000 optimization steps using the Adam optimizer(Kingma et al. 2015) with a learning rate of 0.001. Each experiment was repeated for 50 random seeds. The employed methods are the same as for the nine benchmark tasks, i.e., KP, SKP, and softmax-SKP.

Results We present the average accuracy for all three methods across the evaluated numbers of iterations in Figure 3.18. While the SKP retains an accuracy of almost 100% across all iterations, the KP degrades visibly already from the second iteration. The softmax-SKP retains an average accuracy of 100% up to four iterations, and drops to around 70% for more iterations. In Figure 3.19, we also present the change in the optimization loss for 5000 optimization steps when using eight iterations. All three methods exhibit significantly different behavior, with the loss of the SKP converging to a value close to zero. For the KP, the loss converges to a value around 0.3. For the softmax-SKP, the loss remains constant at its initial value throughout optimization. These insights further confirm that optimization is ineffective for KP and softmax-SKP methods but significantly improved with SKP.

4 Preventing Shared Component Amplification With Multiple Computational Graphs

Many established message-passing neural networks (MPNNs) exhibit shared component amplification (SCA), a phenomenon where applying a single message-passing operation amplifies the same component across all feature channels. This inherently limits the expressiveness of such message-passing steps and the utility of the learnable parameters. Since any message-passing operation using a single computational graph exhibits the SCA property, we propose using multiple computational graphs within each message-passing step and present two methods for deriving such approaches. First, starting with existing MPNNs, we propose splitting the computational graph into multiple edge relations. Second, we define the spectral graph convolution for multiple feature channels and introduce a novel approximation for message-passing that naturally incorporates multiple computational graphs. We prove that both approaches avoid SCA and demonstrate their effectiveness on downstream tasks.

4.1 Introduction

While message-passing neural networks (MPNNs) have achieved promising results, their impact on real-world applications has been limited (Bechler-Speicher et al. 2025). Various phenomena have been identified in the literature that are related to performance limitations (Arnaiz-Rodríguez et al. 2025). These include over-squashing (Alon et al. 2021), under-reaching (Alon et al. 2021; Sun et al. 2022), heterophily (Yan et al. 2022), over-smoothing (Q. Li et al. 2018; Oono et al. 2020), over-correlation (Jin et al. 2022), and rank collapse (Roth and Liebig 2023). However, due to an incomplete theoretical and empirical understanding of these phenomena, modifications lack theoretical grounding, and the relative importance of each phenomenon remains unclear.

We identified shared component amplification (SCA) as one theoretically derived property underlying the observed over-smoothing and rank collapse phenomena, as discussed in Chapter 3. SCA refers to the property that a single message-passing step amplifies the same component for each feature channel and any set of parameters. As a result, such message-passing steps cannot make some feature channels more similar, while simultaneously making others more dissimilar. Unlike over-smoothing, which is typically considered in the iterated or limit case, we identified SCA as a property of a single message-passing operation in Definition 3.1. The property of SCA occurs whenever a single com-

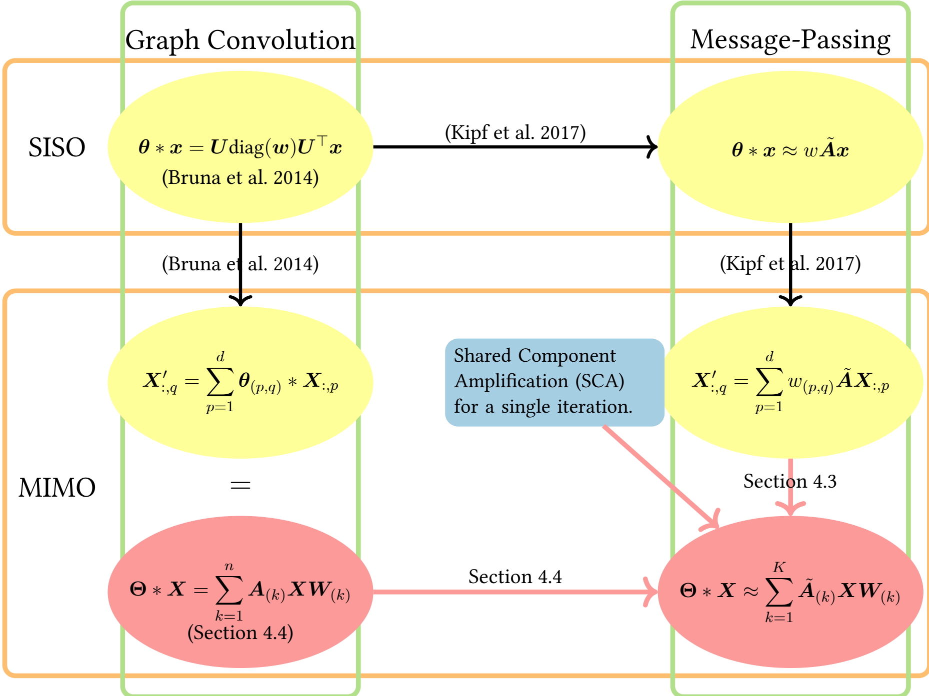


Figure 4.1: Connection between the contributions in this chapter, their connection to Chapter 3, and related work. Yellow parts (■) indicate previous work, blue parts (■) findings from Chapter 3, and pink parts (■) indicate the contributions presented in this chapter.

putational graph is used for message-passing, as shown in Theorem 3.6. While repeatedly applying message-passing steps that exhibit SCA can lead to over-smoothing and rank collapse of the node representations, we focus on a single iteration in this chapter. As shown in Theorem 3.6, a single computational graph always exhibits SCA, and using multiple computational graphs can prevent it, as shown in Proposition 3.12. We will present two frameworks for deriving and utilizing multiple computational graphs. These follow two of our previous publications ((Roth, Bause, et al. 2024) and (Roth and Liebig 2025)). We provide a visual overview in Figure 4.1.

In Section 4.3, we introduce the multi-relational split (MRS) framework, with which we propose to decompose the computational graph of existing MPNNs into multiple edge relations. We propose to do this by assigning each edge to one of multiple relation types. This is motivated by the fact that messages between two nodes may serve a different goal. We prove that the MRS framework can effectively prevent SCA.

In Section 4.4.3, we present a novel derivation of message-passing methods. Instead of approximating MPNNs from the graph convolution in the single-input single-output (SISO) case, as was done with the GCN, we approximate MPNNs directly in the multi-input multi-output (MIMO) case. For that, we first derive the MIMO graph convolution (MIMO-GC) using the convolution theorem and graph Fourier transform. We find that in this multi-channel case, the MIMO-GC naturally utilizes multiple computational graphs, each amplifying a single distinct component. We then approximate the MIMO-GC as localized MIMO-GCs (LMGCs) by localizing the aggregation step. This serves as a general framework with strong theoretical foundations. This framework enables us to study the properties of message-passing methods in a fixed yet flexible form. We prove conditions under which LMGCs prevent SCA and further demonstrate that they allow for injectivity of each message-passing step while applying a linear function.

We empirically evaluate both approaches in Section 4.5, presenting results for the MRS framework in Section 4.5.1 and for the LMGC framework in Section 4.5.2.

4.2 Related Work

We now provide an overview of previous work aimed at avoiding over-smoothing and rank collapse. Specifically, we present approaches related to the SCA property, which we are addressing in this chapter. Based on our proposed solution in Section 3.6.7 of utilizing multiple computational graphs, we also provide an overview of existing methods that change the computational graph for message-passing.

Solutions for Rank Collapse and Over-Smoothing Many different strategies for mitigating rank collapse and over-smoothing have been proposed. To retain information, various methods have been proposed to use residual connections either from the previous iteration (Bresson et al. 2017; G. Li, Xiong, et al. 2020; M. Chen et al. 2020; G. Li, Müller, et al. 2021; Scholkemper et al. 2025) or the initial state (Klicpera et al. 2019; M. Chen et al. 2020; Gu et al. 2020; Roth and Liebig 2022b). Other works have proposed combining the intermediate representations after each iteration into a joint final state (Xu, C. Li, et al. 2018; Fey 2019). Further methods propose to limit the effect of update steps using a gating mechanism (Bresson et al. 2017; Rusch, B. P. Chamberlain, et al. 2023; Dwivedi, Joshi, et al. 2023; Finkelshtein et al. 2024). Other approaches have shown the benefits of applying normalization techniques to node representations (Zhao et al. 2020; G. Li, Xiong, et al. 2020; Scholkemper et al. 2025). Zhou et al. (2021) propose to regularize the Dirichlet energy of the node representations to encourage obtaining less similar representations.

Other lines of work have proposed methods that amplify higher-frequency components rather than low-frequency components using a residual connection (Eliasof, Ruthotto, et al. 2023; Giovanni et al. 2023) using negative edge weights (Bo et al. 2021; Yan et al. 2022), or combining multiple aggregation functions (Corso et al. 2020; Tailor et al. 2022; Rosenbluth et al. 2023). Jin et al. (2022) minimize the correlation between different feature channels in the obtained node representations using regularization.

While these methods can effectively mitigate over-smoothing, they are designed for the multi-iteration message-passing setting. As SCA is a property of a single message-passing iteration, these approaches are not designed to deal with SCA. With our introduction of SCA (Definition 3.1), we can develop targeted strategies to mitigate this property.

Changing the Computational Graph Various approaches have been proposed to modify the provided graph into a more effective computational graph for message passing. Alon et al. (2021) identified that sparsely connected regions lead to the over-squashing of an exponential amount of information into a fixed-size vector. Several works have proposed changing the topology of a graph to reduce sparse connections between regions (Topping et al. 2022; Abboud et al. 2022; Barbero et al. 2024). Other works have found that node representations become similar faster for densely connected regions or graphs (Rong, Huang, et al. 2020; Yan et al. 2022). They have proposed using a sparsified computational graph (Rong, Huang, et al. 2020; Yan et al. 2022; Nguyen et al. 2023; Jamadandi et al. 2024; Rubio-Madrigal et al. 2025). With Co-GNNs, Finkelshtein et al. (2024) propose to discard some of the edges during message passing dynamically.

Several other approaches can be seen as using multiple computational graphs. Some methods were proposed for multi-relational graphs (Schlichtkrull et al. 2018; Vashishth et al. 2020). These works assume a given dataset for which multiple relation types are provided. Butler et al. (2023) propose multigraph neural networks that extend polynomial filters on graphs to given multigraphs. Suresh et al. (2021) propose to add a second edge relation in addition to proximity that is based on the structural similarity of nodes and uses both edge relations for message-passing. Y. Yang et al. (2020) propose to disentangle a graph into multiple factorized graphs, each capturing a distinct latent relation to obtain disentangled node representations. Giunchiglia et al. (2022) propose to utilize an explanability method and discard all representations and edges seen as task-irrelevant. Relatedly, J. Guo et al. (2024) propose the edge splitting GNN that separates available edges of a graph into task-relevant and task-irrelevant ones, producing two computational graphs. Luan et al. (2022) propose an adaptive channel mixing that use the adjacency matrix, the graph Laplacian, and a residual connection as three separate computational graphs to adaptively amplify high- and low-frequency components. Rossi et al. (2023) consider directed graphs and propose to use the graph with reverse edges as a second computational graph. Eliasof, Haber, et al. (2024) propose to utilize distinct edge weights for each feature channel. While all of these works make valuable contributions to MPNNs, the general effect of multiple computational graphs and how they should be designed have not been studied extensively.

4.3 Splitting The Computational Graph of MPNNs

As the first approach to utilizing multiple computational graphs within MPNNs, we now propose a framework that modifies existing MPNNs. The framework starts with any existing message-passing operation and assigns each edge to one of multiple relational types. Depending on the selected relation for an edge, messages are exchanged and transformed differently. This results in a multi-relational graph. This section is based on Roth, Bause, et al. (2024).

4.3.1 Multi-Relational Split MPNNs

We consider an attributed graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ in matrix form with adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and node representations $\boldsymbol{X} \in \mathbb{R}^{n \times d}$. The corresponding set of nodes $\mathcal{V} = \{v_1, \dots, v_n\}$, where v_i corresponds to row i of $\boldsymbol{X}$, and the i-th row and column of $\boldsymbol{A}$. We recall our definition of a message-passing step from Equation (2.17), which obtains an updated state for node v_i as

$$(\text{MPNN}(\boldsymbol{A}, \boldsymbol{X}))_{i,:} = \phi\left(\boldsymbol{X}_{i,:}, \bigoplus \left\{\!\left\{ \alpha_{(i,j)} \cdot \psi\left(\boldsymbol{X}_{i,:}, \boldsymbol{X}_{j,:}\right) \mid v_j \in \mathcal{N}_i \right\}\!\right\}\right), \tag{4.1}$$

where $\psi, \phi \colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ are two functions on tuples, $\oplus$ is a permutation invariant set aggregation function, and $\alpha_{(i,j)} \in \mathbb{R}$ is an edge-specific weighting scalar.

For splitting any such MPNN into a version that utilizes multiple computational graphs, we introduce an edge relation assignment function $f(v_i, v_j) \in [l]$ that assigns each edge (v_i, v_j) of the graph to one of l relation types. Depending on the relation type $k \in [l]$, a different initialization ψ_k of the feature transformation ψ is applied. In general, we define such multi-relational split MPNNs (MRS-MPNNs) as

$$(\text{MRS-MPNN}(\boldsymbol{A}, \boldsymbol{X}))_{i,:} = \phi\left(\boldsymbol{X}_{i,:}, \bigoplus \left\{\!\!\left\{ \alpha_{(i,j)} \cdot \psi_{f(v_i,v_j)}\left(\boldsymbol{X}_{i,:}, \boldsymbol{X}_{j,:}\right) \mid v_j \in \mathcal{N}_i \right\}\!\!\right\}\right). \tag{4.2}$$

Thus, any MPNN of the form given Equation (4.1) can be directly converted into an MRS-MPNN by defining an edge relation assignment function f and instantiating l transformations $\psi_1, \dots, \psi_l$. Before analyzing some of the beneficial properties of MRS-MPNNs, we now provide several examples of established MPNNs and their corresponding MRS-MPNNs. We do not include non-linear activation functions in the following examples, as these are typically applied after applying a message-passing step.

GCN As one of the most commonly employed MPNNs, the graph convolutional network (GCN) applies a linear feature transformation $\boldsymbol{W}$ and aggregates node representations using a symmetric degree normalization. Stated in our MRS framework, the MRS-MPNN can be expressed as

$$(\text{MRS-GCN}(\boldsymbol{A}, \boldsymbol{X}))_{i,:} = \sum_{j \in \mathcal{N}_i} \frac{1}{\sqrt{d_i d_j}} \boldsymbol{W}_{f(v_i,v_j)} \boldsymbol{X}_{j,:} \tag{4.3}$$

where $\boldsymbol{A}_{\text{sym}}$ is the symmetrically normalized adjacency matrix and $\boldsymbol{W}_k \in \mathbb{R}^{d \times d'}$ are independent linear feature transformations for all $k \in [l]$. Apart from the selection of the feature transformation, all other parts remain unchanged from the GCN. Thus, setting all feature transformations $\boldsymbol{W}_k$ to the same matrix would recover the original GCN. This can equivalently be expressed in matrix notation as

$$\text{MRS-GCN}(\boldsymbol{A}, \boldsymbol{X}) = \sum_{k \in [l]} \tilde{\boldsymbol{A}}_{(k)} \boldsymbol{X} \boldsymbol{W}_k, \tag{4.4}$$

where $\left(\tilde{\boldsymbol{A}}_{(k)}\right)_{i,j} = \left(\boldsymbol{A}_{\text{sym}}\right)_{i,j}$ when $f\left(v_i, v_j\right) = k$ and zero otherwise. Equation (4.4) highlights the form that utilizes multiple computational graphs, which can avoid SCA, as outlined in Section 3.6.7.

SAGE Similarly, the SAGE convolution (Hamilton et al. 2017) can be expressed in the MRS framework as

$$\text{MRS-SAGE}(\boldsymbol{A}, \boldsymbol{X}) = \boldsymbol{X}\boldsymbol{V} + \sum_{k \in [l]} \tilde{\boldsymbol{A}}_{(k)} \boldsymbol{X} \boldsymbol{W}_{(k)}\,, \tag{4.5}$$

where $\left(\tilde{\boldsymbol{A}}_{(k)}\right)_{i,j} = (\boldsymbol{A}_{\text{rw}})_{i,j}$ when $f\left(v_i, v_j\right) = k$ and zero otherwise, similar to the MRS-GCN.

GAT The MRS framework can also be applied to adaptively obtained edge weights and multiple attention heads, as used in the graph attention network (GAT) (Velickovic et al. 2018). For $H \in \mathbb{N}$ attention heads, the MRS-GAT is defined as

$$\text{MRS-GAT}(\boldsymbol{A}, \boldsymbol{X}) = \sum_{h \in [H]} \sum_{k \in [l]} \tilde{\boldsymbol{A}}_{(k)}^{(h)} \boldsymbol{X} \boldsymbol{W}_{(k)}^{(h)}\,, \tag{4.6}$$

where $\tilde{\boldsymbol{A}}^{(h)} \in \mathbb{R}^{n \times n}$ is the matrix containing the attention coefficients of head $h \in [H]$, and $\left(\tilde{\boldsymbol{A}}_{(k)}^{(h)}\right)_{i,j} = \left(\tilde{\boldsymbol{A}}^{(h)}\right)_{i,j}$ when $f\left(v_i, v_j\right) = k$ and zero otherwise.

GIN The graph isomorphism network (GIN) (Xu, Hu, et al. 2019) is an established method that enables the injectivity of a message-passing step by utilizing a non-linear multi-layer perceptron (MLP) $\omega\colon \mathbb{R}^d \to \mathbb{R}^d$ as its feature transformation. Converted with the MRS framework, this leads to the update step

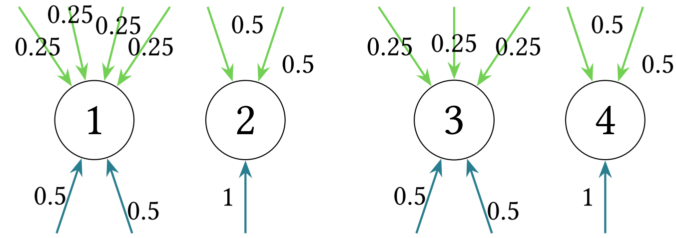
$$\text{MRS-GIN}(\boldsymbol{A}, \boldsymbol{X}) = \sum_{k \in [l]} \tilde{\boldsymbol{A}}_{(k)} \omega_{(k)}(\boldsymbol{X})\,, \tag{4.7}$$

where $\omega_{(k)}\colon \mathbb{R}^d \to \mathbb{R}^d$ is a node-wise (i.e., row-wise) MLP for $k \in [l]$, and $\left(\tilde{\boldsymbol{A}}_{(k)}\right)_{i,j} = (\boldsymbol{A})_{i,j}$ when $f\left(v_i, v_j\right) = k$ and zero otherwise.

GatedGCN The GatedGCN (Dwivedi, Joshi, et al. 2023) extends the GCN by a gating mechanism that allows for incoming messages to be discarded. The corresponding MRS adaptations are given by

$$(\text{MRS-GatedGCN}(\boldsymbol{A}, \boldsymbol{X}))_{i,:} = (\boldsymbol{X}\boldsymbol{V})_{i,:} + \sum_{j \in \mathcal{N}_i} \frac{e_{(i,j)} \odot \boldsymbol{X}_{j,:} \boldsymbol{W}_{f(v_i, v_j)}}{\epsilon + \sum_{k \in \mathcal{N}_i} e_{(i,k)}} \tag{4.8}$$

where $e_{(i,j)} \in \mathbb{R}^d$ is an obtained vector of edge attributes, and $\boldsymbol{V}, \boldsymbol{W}_{(1)}, \ldots, \boldsymbol{W}_{(l)} \in \mathbb{R}^{d \times d}$.



(a) Structurally dependent nodes. **(b)** Structurally independent nodes.

Node	d_1	d_2	SD
1	1	1	✓
2	1	1	✓
3	0.75	1	✗
4	1	1	✓

(c) Numerical dependency.

Figure 4.2: Exemplary visualization of structurally dependent (a) and structurally independent (b) nodes for two computational graphs. Arrows in each color represent the incoming edges for each computational graph. The numbers beside the arrows indicate the edge weights. The numerical dependency (c) indicates the weighted in-degrees of the four considered nodes, and SD indicates whether each node is structurally dependent to node 1.

Theoretical Properties of MRS-MPNNs We now analyze the beneficial properties of MRS-MPNNs. For this analysis, we consider MRS-MPNNs of the form

$$\boldsymbol{X}' = \sum_{k\in[l]} \tilde{\boldsymbol{A}}_{(k)} \boldsymbol{X} \boldsymbol{W}_{(k)} \tag{4.9}$$

where $\tilde{\boldsymbol{A}}_{(k)} \in \mathbb{R}^{n\times n}$ can represent any graph structure and edge weights, and $\boldsymbol{W}_{(k)} \in \mathbb{R}^{d\times d}$ is a linear feature transformation for all $k \in [l]$. This corresponds to multiple computational graphs, as proposed in Section 3.6.7. While we have shown in Proposition 3.12 that this form can generally avoid SCA, we now study the conditions in more detail. Based on the definition of SCA in Definition 3.1, we need to ensure that different components of $\boldsymbol{X}$ can be amplified across distinct feature channels.

To ease our notation, we introduce a vector $\boldsymbol{d}^{(i)}$ that captures the weighted in-degrees for node i and all computational graphs:

Definition 4.1 (Weighted In-Degrees). *Let $\tilde{\boldsymbol{A}}_{(k)} \in \mathbb{R}^{n\times n}$ for $k \in [l]$. For each node $i \in [n]$, the vector $\boldsymbol{d}^{(i)} \in \mathbb{R}^{l}$ contains the weighted in-degrees $d_k^{(i)} = \sum_{m\in[n]} \left(\tilde{\boldsymbol{A}}_{(k)}\right)_{i,m}$ for all edge relations $k \in [l]$.*

This allows us to define the notion of structural dependence between two nodes:

Definition 4.2 (Structural Dependence and Independence). *Two nodes $i, j \in [n]$ are structurally dependent when $\boldsymbol{d}^{(i)} = p \cdot \boldsymbol{d}^{(j)}$ for some $p \in \mathbb{R}$, otherwise they are called structurally independent.*

We visualize an example in Figure 4.2. We can now show that the structural independence of nodes serves as a sufficient condition to avoid SCA:

Theorem 4.1. Let $\boldsymbol{X}' = \sum_{k \in [l]} \tilde{\boldsymbol{A}}_{(k)} \boldsymbol{X} \boldsymbol{W}_{(k)}$ for $\tilde{\boldsymbol{A}}_{(k)} \in \mathbb{R}^{n \times n}$, $\boldsymbol{W}_{(k)} \in \mathbb{R}^{d \times d'}$ for all $k \in [l]$, and $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ be a rank-one matrix. Then, $\boldsymbol{X}'_{i,:}$ and $\boldsymbol{X}'_{j,:}$ are linearly independent for almost every (a.e.) matrices $\boldsymbol{W}_{(1)}, \dots, \boldsymbol{W}_{(l)}$ and a.e. rank-one matrix $\boldsymbol{X}$ if nodes $i \in [n]$ and $j \in [n]$ are structurally independent.

Proof. We require

$$\left(\sum_{k=1}^{l} \tilde{\boldsymbol{A}}_{(k)} \boldsymbol{X} \boldsymbol{W}_{(k)} \right)_{i,:} - c_j \cdot \left(\sum_{k=1}^{l} \tilde{\boldsymbol{A}}_{(k)} \boldsymbol{X} \boldsymbol{W}_{(k)} \right)_{j,:} \neq 0 \tag{4.10}$$

for all choices of $c_j \in \mathbb{R}$. This condition is satisfied for a.e. choice of matrices $\boldsymbol{W}_{(k)}$ for $k \in [l]$ if

$$\left(\tilde{\boldsymbol{A}}_{(k)} \boldsymbol{X} \right)_{i,:} - c_j \cdot \left(\tilde{\boldsymbol{A}}_{(k)} \boldsymbol{X} \right)_{j,:} \neq 0\,. \tag{4.11}$$

for at least one $k \in [l]$. For a.e. rank-one matrix $\boldsymbol{X}$ this inequality further holds whenever

$$\left(\tilde{\boldsymbol{A}}_{(k)} \right)_{i,:} - c_j \cdot \left(\tilde{\boldsymbol{A}}_{(k)} \right)_{j,:} \neq 0\,. \tag{4.12}$$

for at least one $k \in [l]$. Based on the definition of structural independence, this cannot be satisfied for any $c_j \in \mathbb{R}$, and the resulting node representations are linearly independent. □

We note the limitation that Theorem 4.1 requires an a.e. condition on the state $\boldsymbol{X}$. Thus, there exist specific constellations of matrices $\tilde{\boldsymbol{A}}_{(k)}$ so that the resulting representations for nodes i and j remain linearly dependent. However, such matrices $\boldsymbol{X}$ have measure zero, and we will further lift this condition in Theorem 4.4. In all other cases, a single iteration with an MRS-MPNN can amplify distinct components for each feature channel for nodes i and j, preventing SCA. Their node representations can remain similar for one feature channel, while becoming dissimilar for another. The similarity depends on the assigned edge weights, as specific choices can ensure that $\tilde{\boldsymbol{A}}_{(k)} \boldsymbol{X}$ is similar for nodes i and j, while specific other choices can lead to $\tilde{\boldsymbol{A}}_{(k)} \boldsymbol{X}$ being dissimilar for nodes i and j. For which feature channels the representations will be similar or dissimilar then depends on the corresponding feature transformations $\boldsymbol{W}_{(k)}$. As these are typically learnable parameters, the optimization process can effectively control which computational graph should have larger weights for a specific downstream task. This shows that an MRS-MPNN can obtain more informative node representations from a given state for any two structurally independent nodes. As rank-one matrices are not fixed points, such cases also prevent the convergence to a rank-one state. It also means that such functions are injective for structurally independent nodes and a.e. $\boldsymbol{X}$ and $\boldsymbol{W}_{(k)}$, which improves the expressivity of MPNNs. This already highlights a close similarity between avoiding SCA, and as a consequence, rank collapse and over-smoothing, and improving the expressivity to distinguish between structural differences. Two structurally independent nodes have distinct representations after applying one or more iterations of an MRS-MPNN for a.e. $\boldsymbol{X}$. The two node representations remain linearly independent, even after applying a non-linear activation σ, assuming σ is injective and applied component-wise.

Having more structurally independent nodes further allows these nodes to have pairwise linearly independent representations. For a set of structurally independent nodes, each resulting node representation is linearly independent from the set of representations from all other nodes. We formalize this result in the following theorem:

Theorem 4.2. Let p be the number of structurally independent nodes, i.e., for all $i \in [p]$, it holds that $\boldsymbol{d}^{(i)} \neq \sum_{j\in[p],j\neq i} p_j \boldsymbol{d}^{(j)}$ for any $p_j \in \mathbb{R}$. Let σ be a component-wise and injective activation function. Then for $\tilde{\boldsymbol{A}}_{(k)} \in \mathbb{R}^{n\times n}$, a.e. $\boldsymbol{W}_{(k)} \in \mathbb{R}^{d\times d'}$, and a.e. rank-one matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, it holds that

$$\operatorname{rank}\left(\sigma\left(\sum_{k=1}^{l} \tilde{\boldsymbol{A}}_{(k)}\boldsymbol{X}\boldsymbol{W}_{(k)}\right)\right) \geq p\,. \tag{4.13}$$

Proof. For notational simplicity, we assume the first nodes $[p]$ to be structurally independent. We then require

$$\left(\sigma\left(\sum_{k=1}^{l} \tilde{\boldsymbol{A}}_{(k)}\boldsymbol{X}\boldsymbol{W}_{(k)}\right)\right)_{i,:} - \sum_{j\in[p],j\neq i} c_j \left(\sigma\left(\sum_{k=1}^{l} \tilde{\boldsymbol{A}}_{(k)}\boldsymbol{X}\boldsymbol{W}_{(k)}\right)\right)_{j,:} \neq 0 \tag{4.14}$$

for all choices of $c_j \in \mathbb{R}, j \in [p], j \neq i$. This condition is satisfied for a.e. choice of matrices $\boldsymbol{W}_{(k)}$ for $k \in [l]$ if

$$\left(\sum_{k=1}^{l} \tilde{\boldsymbol{A}}_{(k)}\boldsymbol{X}\boldsymbol{W}_{(k)}\right)_{i,:} - \sum_{j\in[p],j\neq i} c_j \left(\sum_{k=1}^{l} \tilde{\boldsymbol{A}}_{(k)}\boldsymbol{X}\boldsymbol{W}_{(k)}\right)_{j,:} \neq 0\,. \tag{4.15}$$

Furthermore, for a.e. choice of matrices $\boldsymbol{W}_{(k)}$ for $k \in [l]$ this inequality holds whenever

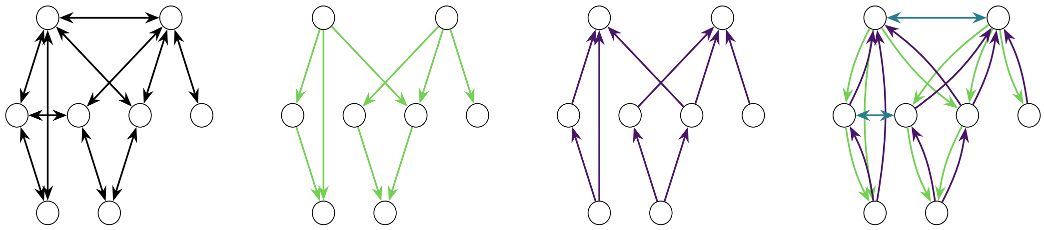
$$\left(\tilde{\boldsymbol{A}}_{(k)}\boldsymbol{X}\right)_{i,:} - \sum_{j\in[p],j\neq i} c_j \left(\tilde{\boldsymbol{A}}_{(k)}\boldsymbol{X}\right)_{j,:} \neq 0\,. \tag{4.16}$$

for at least one $k \in [l]$. For a.e. rank-one matrix $\boldsymbol{X}$ this inequality further holds whenever

$$\left(\tilde{\boldsymbol{A}}_{(k)}\right)_{i,:} - \sum_{j\in[p],j\neq i} c_j \left(\tilde{\boldsymbol{A}}_{(k)}\right)_{j,:} \neq 0\,. \tag{4.17}$$

for at least one $k \in [l]$. Based on the definition of structural independence, this cannot be satisfied for any $c_j \in \mathbb{R}$, and the resulting node representations are linearly independent.

As this holds equivalently for any of the p structurally independent nodes, this results in p linearly independent representations, implying a minimal rank of p. $\square$



(a) Original graph. **(b)** DAR. **(c)** Reverse DAR. **(d)** MRS-MPNN$_{\text{DA}}$.

Figure 4.3: Different computational graphs described for MRS-MPNN$_{\text{DA}}$. The original graph is represented in (a), the obtained directed acyclic relation (DAR) using the node-degree as a partial ordering in (b), and the reverse DAR using the reverse partial ordering in (c). In (d), these two computational graphs are combined with the relation covering all other edges of the original graph.

This aligns with the goal of constructing a message-passing step that does not exhibit SCA. When we have p structurally independent nodes, we can amplify distinct components for at least p feature channels, and the learnable parameters given by $\boldsymbol{W}_{(k)}$ control which components get amplified for which feature channel. This allows the learnable parameters to be more effective and the optimization to be more impactful. Adding feature channels can also capture new combinations of amplified components. When linearly independent representations are desired, maximizing the number of structurally independent nodes helps achieve this goal.

4.3.2 Obtaining Multiple Relations Using a Partial Ordering

Based on the general MRS framework and the theoretical properties we presented in Theorem 4.1 and Theorem 4.2, an MRS-MPNN can improve the obtained representations and enable learnable parameters to be more effective. It remains open how the edge relation assignment function $f(v_i, v_j)$ can be defined. Ideally, such a function f ensures the existence of structurally independent nodes. Finding the optimal function f is highly task-dependent, and prior knowledge can be leveraged in many cases. For many applications, different edge relations can be naturally constructed, as nodes and edges correspond to distinct objects. For example, in traffic networks, roads leading to highways could be assigned to a different relation than nodes within a residential area. For chemical data, the type of bond could determine the relation type. We envision considerable potential for future work to identify specific edge splits for tasks of interest.

Here, we propose a generic option that utilizes graph-theoretical properties. Consider the following motivational example. A multi-head attention-based MPNN can also be viewed as a graph with multiple relations between all connected nodes. Attention normalizes edge weights to sum to one for each node within one computational graph, and all edge weights are positive. Graphs are typically ergodic, which means that they are strongly connected, i.e., a path exists between all pairs of nodes, and aperiodic, i.e., the length of their cycles does not have a common divisor > 1. Such graphs are unable to produce structurally independent nodes:

Proposition 4.1. Let $\tilde{\boldsymbol{A}}_{(k)} \in \mathbb{R}^{n \times n}$ represent an ergodic graph for all $k \in [l]$ such that $\sum_{j \in [n]} \left(\tilde{\boldsymbol{A}}_{(k)}\right)_{i,j} = 1$ for all $i \in [n]$ and $\left(\tilde{\boldsymbol{A}}_{(k)}\right)_{i,j} \geq 0$ for all $i, j \in [n]$. Then, any pair of nodes is structurally dependent.

Proof. For every relation $k \in [l]$ and every node $i \in [n]$, we have $\sum_{j \in [n]} \left(\tilde{\boldsymbol{A}}_{(k)}\right)_{i,j} = 1$. This is a contradiction to the definition of structural independence, which requires $\sum_{j \in [n]} \left(\boldsymbol{A}_{(k)}\right)_{p,j} \neq \sum_{j \in [n]} \left(\boldsymbol{A}_{(k)}\right)_{q,j}$ for some $p \neq q$. □

We note the importance of the ergodic assumption. For a graph that is not ergodic, the same cannot be stated. A graph that does not contain any ergodic subgraphs is a directed acyclic graph (DAG), which we define for a relation as follows:

Definition 4.3 (Directed Acyclic Relation (DAR)). *A relation $\tilde{\boldsymbol{A}}$ is called a directed acyclic relation (DAR) if there exists a strict partial ordering $\prec$ on the nodes $i, j \in [n]$ such that all edges $\tilde{A}_{i,j} \neq 0$, satisfy $i \prec j$.*

We further refer to a node with no incoming edges, i.e., $\tilde{\boldsymbol{A}}_{i,:} = \boldsymbol{0} \in \mathbb{R}^n$, as a root node in $\tilde{\boldsymbol{A}}$. We find that for two DARs with distinct root nodes, these nodes are always structurally independent:

Proposition 4.2. Let $\tilde{\boldsymbol{A}}_{(1)}, \tilde{\boldsymbol{A}}_{(2)} \in \mathbb{R}^{n \times n}$ be two DARs with distinct root nodes. Then, $\tilde{\boldsymbol{A}}_{(1)}$ and $\tilde{\boldsymbol{A}}_{(2)}$ have structurally independent nodes.

Proof. Let i be a root node of $\tilde{\boldsymbol{A}}_{(1)}$ that is not a root node of $\tilde{\boldsymbol{A}}_{(2)}$. Then $\sum_{j \in [n]} \left(\tilde{A}_{(1)}\right)_{i,j} = 0$ and $\sum_{j \in [n]} \left(\tilde{A}_{(2)}\right)_{i,j} \neq 0$. For any other node $k \in [n]$, we have $\sum_{j \in [n]} \left(\tilde{A}_{(1)}\right)_{k,j} \neq 0$, which implies structural independence. □

Constructing two DARs with distinct root nodes is one way to ensure structurally independent nodes, which then allows linearly independent node representations. Transforming a given graph $G = (\boldsymbol{A}, \boldsymbol{X})$ into a directed acyclic graph is a well-known algorithmic challenge, which arises in topological sorting (Cormen et al. 1989) and finding feedback arc sets (Garey et al. 1979). Instead, based on our Definition 4.3, we start by defining a strict partial ordering $\prec$ on the nodes and only utilize the edges that satisfy the relation. Starting with the original aggregation function $\tilde{\boldsymbol{A}}$, this results in the first DAR with entries $\left(\tilde{\boldsymbol{A}}_{(1)}\right)_{i,j} = \tilde{\boldsymbol{A}}_{i,j}$ when $i \prec j$ and zero otherwise. We obtain the second DAR, as the reverse DAR using the reverse strict partial ordering, i.e., the entries are $\left(\tilde{\boldsymbol{A}}_{(2)}\right)_{i,j} = \tilde{\boldsymbol{A}}_{i,j}$ for $j \prec i$ and zero otherwise. As we employ a partial ordering, some nodes may not be related by $\prec$. For these remaining edges, we propose a third edge relation $\tilde{\boldsymbol{A}}_{(3)}$. Their combination results in the original adjacency matrix, i.e., $\tilde{\boldsymbol{A}} = \tilde{\boldsymbol{A}}_{(1)} + \tilde{\boldsymbol{A}}_{(2)} + \tilde{\boldsymbol{A}}_{(3)}$. We define the corresponding edge relation assignment function as

$$f(i,j) = \begin{cases} 1, & \text{if } i \prec j \\ 2, & \text{if } j \prec i \\ 3, & \text{otherwise} \,. \end{cases} \tag{4.18}$$

Figure 4.3 visualizes various computational graphs and edge relations. We refer to this instantiation of the MRS framework applied to an MPNN as MRS-MPNN$_{DA}$ due to utilizing multiple directed acyclic relations. While we cannot guarantee that the third graph leads to more structurally independent nodes, additional graphs cannot hurt the number of structurally independent nodes.

To apply the MRS-MPNN$_{DA}$, we additionally need to define the strict partial ordering $\prec$. The partial ordering needs to align with the considered task. While any method ensures structurally independent nodes, the edges within one relation type should be similar for the optimization process to obtain suitable parameters more effectively. When the messages sent across edges from different edge relations benefit from similar feature transformation, the gain of the MRS framework will also be less significant. Graph traversal algorithms or centrality measures can provide such an ordering. We propose to use the node degree as an efficient and general ordering. Messages sent from higher-degree nodes to lower-degree nodes differ from those sent from lower-degree nodes to higher-degree nodes for specific tasks. For molecular data, the degree of a node corresponds to other atom types than those of lower degree (Wells 2012). Sending messages based on relations between atom types may be promising. As the choice of edge relation assignment function has a significant impact on node representations and the optimization process, this allows future methods to investigate which split specifically makes sense for a given task and which prior knowledge may be beneficial to incorporate for that task.

4.4 What Can We Learn From MIMO Graph Convolutions?

The MRS framework converts the computational graph of a given MPNN into multiple computational graphs, allowing MPNNs to avoid SCA while maintaining other proposed benefits. We now propose a novel derivation of message-passing methods from the spectral graph convolution. This will not only demonstrate the close connection between utilizing multiple computational graphs in MPNNs but also provide a more general framework for designing future message-passing methods. This section is based on Roth and Liebig (2025).

4.4.1 SISO Graph Convolution

We first provide an overview of related work on the definition of graph convolution and its approximations. We visualize the key elements of this overview in Figure 4.4. Initially, the single-input single-output case (SISO) was considered where the node representations $\boldsymbol{x} \in \mathbb{R}^n$ consist of a single feature for each node. This is mapped to updated node representations $\boldsymbol{x}' \in \mathbb{R}^n$ that also consist of a single feature for each node. Bruna et al. (2014) defined the SISO graph convolution as

$$\boldsymbol{\theta} * \boldsymbol{x} = \boldsymbol{U}\mathrm{diag}(\boldsymbol{w})\boldsymbol{U}^\top \boldsymbol{x} \tag{4.19}$$

where $\boldsymbol{w} = \boldsymbol{U}^\top \boldsymbol{\theta} \in \mathbb{R}^n$ using the graph Fourier transform $\boldsymbol{U}^\top$ as the Fourier transform F within the convolution theorem. We refer to our introduction of the graph Fourier transform in Section 2.2.2 and the graph convolution in Section 2.3.2 for more details. Most node representations $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ have some number d of features for each node. Bruna et al. (2014) propose to extend the SISO graph convolution to this multi-input multi-output (MIMO) case by applying a separate SISO graph convolution to every combination of input and output feature channels, i.e., the q-th feature channel of the resulting node representations $\boldsymbol{X}' \in \mathbb{R}^{n \times c}$ is updated as

$$\boldsymbol{X}'_{:,q} = \sum_{p=1}^{d} \boldsymbol{\theta}_{(p,q)} * \boldsymbol{X}_{:,p}\,, \tag{4.20}$$

where each $\boldsymbol{\theta}_{(p,q)} \in \mathbb{R}^n$ is a vector for every $p \in [d]$ and $q \in [c]$. Hammond et al. (2011) propose Chebyshev polynomials that approximate the SISO graph convolution. Sandryhaila et al. (2013) propose polynomial filters for graphs in the SISO case more generally. For polynomial approximations, the updated state $\boldsymbol{x}' \in \mathbb{R}^n$ is set as

$$\boldsymbol{x}' = \sum_{k=0}^{K} w^{(k)} \boldsymbol{A}_{\text{sym}}^k \boldsymbol{x} \tag{4.21}$$

where $w^{(k)} \in \mathbb{R}$ are the polynomial coefficients and the symmetrically normalized adjacency matrix $\boldsymbol{A}_{\text{sym}}$ is the variable of the polynomial. Defferrard et al. (2016) propose to extend polynomial filters to the MIMO case equivalently to the extension of SISO graph convolutions by applying a polynomial filter to each combination of input and output feature channel, i.e.,

$$\boldsymbol{X}'_{:,q} = \sum_{p=1}^{d} \sum_{k=0}^{K} w_{(p,q)}^{(k)} \boldsymbol{A}_{\text{sym}}^k \boldsymbol{X}_{:,p} \tag{4.22}$$

where each $w_{(p,q)}^{(k)} \in \mathbb{R}$ is a coefficient for each p, q, k. Gama et al. (2018) similarly extend polynomial filters to the MIMO case. Kipf et al. (2017) propose the GCN as a localized first-order approximation of polynomial filters on graphs.

$$\boldsymbol{x}' = w \boldsymbol{A}_{\text{sym}} \boldsymbol{x}\,, \tag{4.23}$$

where $w \in \mathbb{R}$ is the single parameter of this approach. The GCN adds self-loops to the adjacency matrix before normalization, which we omit here for notational simplicity. Equivalently to Bruna et al. (2014), Kipf et al. (2017) also extend the GCN to the MIMO

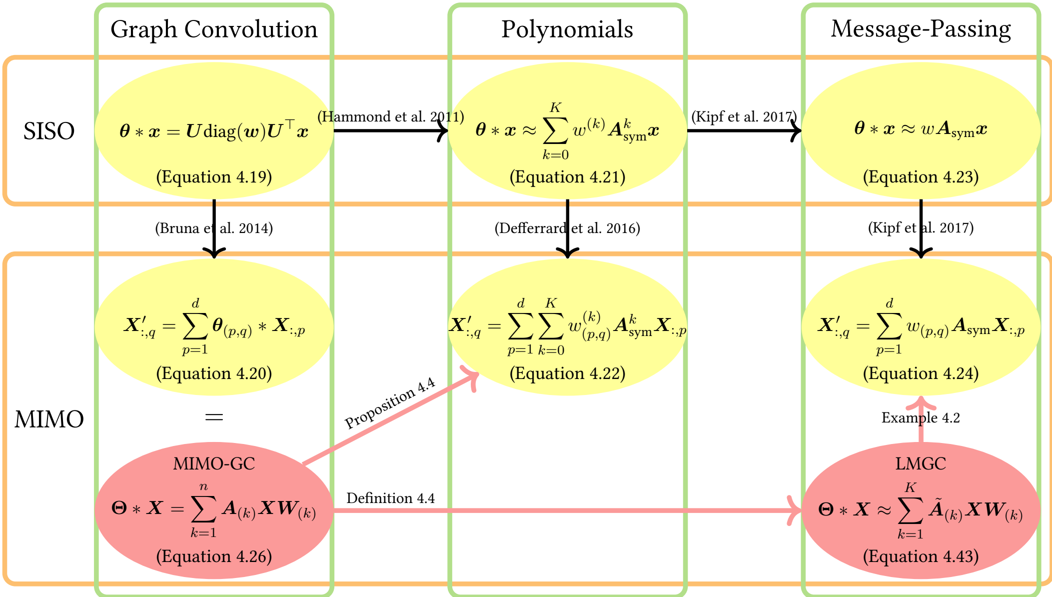


Figure 4.4: Connection between the MIMO-GC, the LMGC and previous work on defining the graph convolution, polynomial, and localized approximations in the SISO and the MIMO case. Yellow parts ( ) indicate previous work, while pink parts ( ) indicate our contributions.

case by applying the approximation to each combination of input and output channel, i.e.,

$$\boldsymbol{X}'_{:,q} = \sum_{p=1}^{d} w_{(p,q)} \boldsymbol{A}_{\text{sym}} \boldsymbol{X}_{:,p} \tag{4.24}$$

for each $q \in [c]$ and $w_{(p,q)} \in \mathbb{R}$ for all p, q. Subsequently, many other MPNNs have been proposed as variations of the GCN (Velickovic et al. 2018; Xu, Hu, et al. 2019; Roth and Liebig 2022b). This highlights that methods such as GCN and its modifications approximate the graph convolution in the SISO case, but are applied in the MIMO case. We will propose an approximation that avoids this indirect route.

4.4.2 MIMO Graph Convolution

Instead of approximating the SISO graph convolution, we propose to perform the approximation directly in the MIMO case. For that, we first need to define the graph convolution in the MIMO case. Given a node representations matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, we want to perform the graph convolution in the Fourier domain

$$(\boldsymbol{\Theta} * \boldsymbol{X})_{i,:} = F^{-1}(F(\boldsymbol{\Theta}) \boxdot F(\boldsymbol{X})) \tag{4.25}$$

based on the convolution theorem (O'Neil 1963), where $\boxdot$ is the node-wise multiplication $(F(\boldsymbol{\Theta}) \boxdot F(\boldsymbol{X}))_{i,:} = F(\boldsymbol{\Theta})_{i,:,:} \cdot F(\boldsymbol{X})_{i,:}$. Following Bruna et al. (2014), we define the

Fourier transform F as the graph Fourier transform given by the eigenvectors of $\boldsymbol{A}_{\text{sym}}$. The convoluted node representation matrix $\boldsymbol{X}' = \boldsymbol{\Theta} * \boldsymbol{X} \in \mathbb{R}^{n \times c}$ can have $c \in \mathbb{N}$ feature channels for every node. The filter $\boldsymbol{\Theta} \in \mathbb{R}^{n \times c \times d}$ needs to contain the element-wise mapping from d to c feature channels $\boldsymbol{\Theta}_{i,:,:} \in \mathbb{R}^{c \times d}$. Based on these definitions, we obtain the MIMO graph convolution as follows:

Theorem 4.3 (MIMO Spectral Graph Convolution (MIMO-GC)). Let $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{\Theta} \in \mathbb{R}^{n \times c \times d}$ and the Fourier transform $F(\boldsymbol{M}) := \boldsymbol{U}^\top \times_1 \mathbf{M}$ be defined using mode-1 multiplication $\times_1$ and the eigenvector matrix $\boldsymbol{U} \in \mathbb{R}^{n \times n}$ of $\boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{n \times n}$ for any tensor $\mathbf{M} \in \mathbb{R}^{n \times d_1 \times \dots \times d_k}$ for any $d_1, \dots, d_k, k \in \mathbb{N}$. Then,

$$(\boldsymbol{\Theta} * \boldsymbol{X})_{i,:} = \left(\sum_{k=1}^{n} \boldsymbol{A}_{(k)} \boldsymbol{X} \boldsymbol{W}_{(k)} \right)_{i,:} \tag{4.26}$$

$$= \sum_{j=1}^{n} \boldsymbol{X}_{j,:} \boldsymbol{W}_{(i,j)} \tag{4.27}$$

where $\boldsymbol{A}_{(k)} = \boldsymbol{U}_{:,k} \boldsymbol{U}_{:,k}^\top \in \mathbb{R}^{n \times n}$, $\boldsymbol{W}_{(k)} = \left(\boldsymbol{U}^\top \times_1 \boldsymbol{\Theta} \right)_{k,:,:}^\top \in \mathbb{R}^{d \times c}$ and $\boldsymbol{W}_{(i,j)} = \left(\sum_{k=1}^{n} U_{i,k} U_{j,k} \boldsymbol{W}_{(k)} \right)^\top \in \mathbb{R}^{c \times d}$.

Proof. We first apply the graph Fourier transform F to $\boldsymbol{\Theta}$ and $\boldsymbol{X}$ using the mode-1 multiplication and $\boldsymbol{U}^\top$. For $\boldsymbol{\Theta}$, we get the graph Fourier transformed filter

$$\hat{\mathbf{W}}_{:,p,q} = \left(\boldsymbol{U}^\top \times_1 \boldsymbol{\Theta} \right)_{:,p,q} = \boldsymbol{U}^\top \boldsymbol{\Theta}_{:,p,q} \tag{4.28}$$

with $\hat{\mathbf{W}} \in \mathbb{R}^{n \times c \times d}$, and for $\boldsymbol{X}$, we have

$$\hat{\boldsymbol{X}} = F(\boldsymbol{X}) = \boldsymbol{U}^\top \boldsymbol{X} \,. \tag{4.29}$$

Based on this, we get

$$\boldsymbol{\Theta} * \boldsymbol{X} = \boldsymbol{U} \left(\hat{\mathbf{W}} \boxdot \boldsymbol{U}^\top \boldsymbol{X} \right) , \tag{4.30}$$

using the inverse graph Fourier transform $F^{-1}(\boldsymbol{M}) = \boldsymbol{U} \boldsymbol{M}$ for any $\boldsymbol{M}$ of suitable shape. Similarly to the SISO graph convolution, we replace the node-wise product $\boxdot$ with a matrix multiplication by diagonalizing $\hat{\mathbf{W}}$ as the block matrix of diagonal blocks in the matrix

$$\boldsymbol{E} = \begin{bmatrix} \hat{W}_{1,1,1} & 0 & 0 & & \hat{W}_{1,1,d} & 0 & 0 \\ 0 & \ddots & 0 & \dots & 0 & \ddots & 0 \\ 0 & 0 & \hat{W}_{n,1,1} & & 0 & 0 & \hat{W}_{n,1,d} \\ & \vdots & & & & \vdots & \\ \hat{W}_{1,c,1} & 0 & 0 & & \hat{W}_{1,c,d} & 0 & 0 \\ 0 & \ddots & 0 & \dots & 0 & \ddots & 0 \\ 0 & 0 & \hat{W}_{n,c,1} & & 0 & 0 & \hat{W}_{n,c,d} \end{bmatrix} \in \mathbb{R}^{nc \times nd} \,. \tag{4.31}$$

This form allows us to simplify the vectorized MIMO graph convolution as

$$\operatorname{vec}(\boldsymbol{\Theta} * \boldsymbol{X}) = (\boldsymbol{I}_c \otimes \boldsymbol{U})\, \boldsymbol{E} \operatorname{vec}\left(\boldsymbol{U}^\top \boldsymbol{X}\right) . \tag{4.32}$$

We can further decompose $\boldsymbol{E}$ as a sum of Kronecker products as

$$\boldsymbol{E} = \sum_{k=1}^{n} \hat{\boldsymbol{W}}_{k,:,:} \otimes \boldsymbol{I}_n^{(k)} , \tag{4.33}$$

where $\boldsymbol{I}_n^{(k)} \in \mathbb{R}^{n \times n}$ contains a single one at position (k, k) and zeros at all other positions. Substituting this decomposition in Equation (4.32), we obtain

$$\operatorname{vec}(\boldsymbol{\Theta} * \boldsymbol{X}) = \left(\sum_{k=1}^{n} \hat{\boldsymbol{W}}_{k,:,:} \otimes \boldsymbol{U}_{:,k} \boldsymbol{U}_{:,k}^\top\right) \operatorname{vec}(\boldsymbol{X}) . \tag{4.34}$$

Based on the property $(\boldsymbol{A} \otimes \boldsymbol{B}) \operatorname{vec}(\boldsymbol{C}) = \boldsymbol{B}\boldsymbol{C}\boldsymbol{A}^\top$ of the Kronecker product for matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ of suitable shape, we state the MIMO graph convolution in the original matrix form as

$$\boldsymbol{\Theta} * \boldsymbol{X} = \sum_{k=1}^{n} \boldsymbol{U}_{:,k} \boldsymbol{U}_{:,k}^\top \boldsymbol{X} \hat{\boldsymbol{W}}_{k,:,:}^\top . \tag{4.35}$$

Defining $\boldsymbol{A}_{(k)} = \boldsymbol{U}_{:,k} \boldsymbol{U}_{:,k}^\top$ and $\boldsymbol{W}_{(k)} = \hat{\boldsymbol{W}}_{k,:,:}$ concludes the proof. □

Due to the generality of the MIMO-GC, approximations of the graph convolution applied in the MIMO case should closely mimic the MIMO-GC. The MIMO-GC can be equivalently expressed as applying the SISO graph convolution for every combination of input and output feature channel, as given in Equation (4.20). Knowing the general form of the graph convolution for multiple feature channels allows us to study its properties. These properties can then be used to improve approximations.

The first observation is that the MIMO-GC utilizes n multiple computational graphs $\boldsymbol{A}_{(k)} \in \mathbb{R}^{n \times n}$, each being a rank-one matrix $\boldsymbol{A}_{(k)} = \boldsymbol{U}_{:,k} \boldsymbol{U}_{:,k}^\top$. Thus, each such computational graph can only amplify exactly one component in the graph that corresponds to the eigenvector $\boldsymbol{U}_{:,k}$ of $\boldsymbol{A}_{\text{sym}}$. The corresponding feature transformation $\boldsymbol{W}_{(k)} \in \mathbb{R}^{d \times c}$ encodes how much this component is amplified from each of the d input feature channels for each of the c output feature channels. This aligns well with our findings from Chapter 3, particularly our definition of shared component amplification (SCA). In Definition 3.1, we have shown that a single computational graph can only amplify the same component maximally for all feature channels jointly. Each computational graph $\boldsymbol{A}_{(k)}$ is orthogonal, meaning that it also amplifies independent components in the data. By utilizing multiple computational graphs, the MIMO-GC allows for the amplification of arbitrary components for each feature channel independently. We formally show the universality of the MIMO-GC to map almost any given node representations $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ to any desirable node representation $\boldsymbol{X}' \in \mathbb{R}^{n \times c}$ in the following proposition:

Proposition 4.3. Let $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ and $\boldsymbol{X}' \in \mathbb{R}^{n\times c}$. Then, there exists a $\boldsymbol{\Theta} \in \mathbb{R}^{n\times c\times d}$ such that

$$\boldsymbol{\Theta} * \boldsymbol{X} = \boldsymbol{X}' \tag{4.36}$$

for almost every $\boldsymbol{X}$.

Proof. We decompose $\boldsymbol{X}$ and $\boldsymbol{X}'$ as a sum of rank-one matrices, each with all columns equal to a scaled version of an eigenvector $\boldsymbol{U}_{:,k} \in \mathbb{R}^{n\times 1}$ for all $k \in [n]$. For $\boldsymbol{X}'$, we obtain the decomposition

$$\boldsymbol{X}' = \sum_{k=1}^{n} \boldsymbol{U}_{:,k}\boldsymbol{c}_{(k)}^{\top} \tag{4.37}$$

for some $\boldsymbol{c}_{(k)} \in \mathbb{R}^{c}$ for all $k \in [n]$. Similarly, we use the decomposition

$$\boldsymbol{X} = \sum_{k=1}^{n} \boldsymbol{U}_{:,k}\boldsymbol{d}_{(k)}^{\top} \tag{4.38}$$

for some $\boldsymbol{d}_{(k)} \in \mathbb{R}^{d}$ for all $k \in [n]$. Inserting this for the MIMO-GC, we obtain

$$\boldsymbol{\Theta} * \boldsymbol{X} = \sum_{k=1}^{n} \boldsymbol{U}_{:,k}\boldsymbol{d}_{(k)}^{\top}\boldsymbol{W}_{(k)} \,. \tag{4.39}$$

By defining $\boldsymbol{W}_{(k)}$ so that $\boldsymbol{c}_{(k)} = \boldsymbol{d}_{(k)}\boldsymbol{W}_{(k)}$ which is possible for all $\boldsymbol{d}_{(k)} \neq \boldsymbol{0}$, we conclude the proof. □

This confirms the universal power of the MIMO-GC to represent arbitrary such linear functions. The second interpretation of the MIMO-GC uses the form given by Equation (4.27). For every combination of nodes i and j, a distinct feature transformation $\boldsymbol{W}_{(i,j)}$ is applied. While these feature transformations are intertwined through $\boldsymbol{W}_{(k)}$, this form highlights the benefits of applying distinct feature transformations $\boldsymbol{W}_{(i,j)}$ for every message between pairs of nodes. As Proposition 4.3 holds for this equivalent form, applying distinct feature transformations improves the ability of the MIMO-GC to represent arbitrary functions and amplify distinct components between every combination of input and output channel. For example, this form helps with understanding the benefits of recent MPNNs that have proposed the use of edge-wise distinct feature transformations, e.g., in neural sheaf diffusion (Hansen et al. 2020; Bodnar et al. 2022).

While obtaining the MIMO-GC is valuable for gaining a better understanding and intuition for applying MPNNs in the MIMO case, the MIMO-GC will typically not be applied directly for the same reasons that the SISO-GC is not applied. First, the graph Fourier transform is graph-dependent, so applying a filter $\boldsymbol{\Theta}$ to a different graph is typically not reasonable. In addition, the computational complexity of obtaining the graph Fourier transform and applying it is infeasible for many graphs. The graph Fourier transform requires obtaining all eigenvectors of $\boldsymbol{A}_{\text{sym}}$ and applying n dense matrices, in addition to applying n feature transformations $\boldsymbol{W}_{(k)}$, which has a total runtime complexity of $\mathcal{O}(n^2 \cdot c \cdot d)$.

How Can We Benefit From the MIMO-GC? Instead of directly applying the MIMO-GC, we aim to use it as a starting point to derive approximations and other methods, similar to how the SISO-GC is utilized. Understanding existing approaches for MIMO computation helps design novel approaches. For example, we show in the following that any polynomial MIMO filter can be represented as a MIMO-GC with constraints on the feature transformation matrices:

Proposition 4.4. Let $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, $P \in \mathbb{N}$, and $\boldsymbol{V}_{(p)} \in \mathbb{R}^{d\times c}$ for $0 \leq p \leq P$ and any $d, c \in \mathbb{N}$. Then, there exists a filter $\boldsymbol{\Theta} \in \mathbb{R}^{n\times c\times d}$ such that

$$\sum_{p=0}^{P} \boldsymbol{A}_{\text{sym}}^{p} \boldsymbol{X} \boldsymbol{V}_{(p)} = \boldsymbol{\Theta}_{\text{poly}} * \boldsymbol{X} \,. \tag{4.40}$$

Proof. Based on the eigendecomposition $\boldsymbol{A}_{\text{sym}}^{p} = \sum_{k=1}^{n} \boldsymbol{U}_{:,k} \boldsymbol{U}_{:,k}^{\top} \lambda_k^p$ and the definition of the MIMO-GC, we obtain

$$\sum_{p=0}^{P} \boldsymbol{A}_{\text{sym}}^{p} \boldsymbol{X} \boldsymbol{V}_{(p)} = \sum_{k=1}^{n} \sum_{p=0}^{P} \boldsymbol{U}_{:,k} \boldsymbol{U}_{:,k}^{\top} \lambda_k^p \boldsymbol{X} \boldsymbol{V}_{(p)} \tag{4.41}$$

By defining $\boldsymbol{W}_{(k)} = \sum_{p=0}^{P} \lambda_k^p \boldsymbol{V}_{(p)}$, we obtain the form of the MIMO-GC. $\square$

This further confirms the expressive power of the MIMO-GC. As an example, consider the update step from the GCN as a first-order polynomial that can be expressed as MIMO-GC in the following form:

Example 4.1. Let $\boldsymbol{X}' = \boldsymbol{A}_{\text{sym}} \boldsymbol{X} \boldsymbol{W}$ for some $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, $\boldsymbol{W} \in \mathbb{R}^{d\times c}$, $\boldsymbol{W} \in \mathbb{R}^{d\times c}$, and $\boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{n\times n}$ for $n, d, c \in \mathbb{N}$. Then,

$$\boldsymbol{X}' = \sum_{k=1}^{n} \boldsymbol{U}_{:,k} \boldsymbol{U}_{:,k}^{\top} \boldsymbol{X} \boldsymbol{W}_{(k)} \tag{4.42}$$

where $\boldsymbol{W}_{(k)} = \lambda_k \boldsymbol{W}$ and λ_k is the eigenvalue of $\boldsymbol{A}_{\text{sym}}$ corresponding to eigenvector $\boldsymbol{U}_{:,k}$.

As such, the GCN is equal to a MIMO-GC, which applies the same feature transformation to all components, with the only difference being that all components are then scaled by their respective eigenvalue of $\boldsymbol{A}_{\text{sym}}$ jointly across all feature channels. This also serves as an additional intuitive explanation for the SCA phenomenon, as the components are scaled according to the aggregation matrix $\boldsymbol{A}_{\text{sym}}$. The feature transformation cannot scale one component more than another. This similarly holds if any other single computational graph $\tilde{\boldsymbol{A}}$ was used, as the feature transformation is always jointly applied to all components with the fixed scaling of all feature channels only depending on the spectrum of $\tilde{\boldsymbol{A}}$. For $\boldsymbol{A}_{\text{sym}}$, we have $\boldsymbol{U}_{:,1} = \boldsymbol{D}^{1/2}\mathbf{1}$ which will be amplified maximally as it corresponds to the maximal eigenvalue $\lambda_1 = 1$ of $\boldsymbol{A}_{\text{sym}}$. This leads to the well-known phenomenon of over-smoothing (Oono et al. 2020).

Apart from gaining a better understanding of existing approaches by relating them to the MIMO-GC, we can also design novel methods that share similar beneficial properties to the MIMO-GC. To amplify multiple components, such as approximations of the MIMO-GC, multiple computational graphs should be utilized. The learnable feature transformations can then control which components get amplified for each feature channel. Equivalently, we have shown that methods can employ distinct feature transformations between all pairs of nodes. Such a formulation enables the learnable parameters to be more effective, which in turn facilitates optimization and yields more informative node representations.

4.4.3 Localized MIMO Graph Convolutions

Following Equation (4.27), the MIMO-GC applies a distinct feature transformation between any pair of nodes i and j. While the MIMO-GC utilizes a fully connected graph, we propose a localized approximation that applies a distinct feature transformation between every connected pair of nodes. This is equivalent to utilizing multiple computational graphs, as shown for the MIMO-GC (Theorem 4.3). Formally, we define the localized MIMO-GC (LMGC) as follows:

Definition 4.4 (Localized MIMO Graph Convolution (LMGC)). *Let $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ be a graph in matrix form with $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{X} \in \mathbb{R}^{n\times d}$. A localized MIMO graph convolution (LMGC) is a permutation equivariant function $f(\boldsymbol{A}, \boldsymbol{X}) \in \mathbb{R}^{n\times c}$ of the form*

$$\begin{aligned} (f(\boldsymbol{A}, \boldsymbol{X}))_{i,:} &= \sum_{j\in N_i} \boldsymbol{X}_{j,:}\tilde{\boldsymbol{W}}_{(i,j)} \\ &= \left(\sum_{k\in[K]} \tilde{\boldsymbol{A}}_{(k)}\boldsymbol{X}\boldsymbol{W}_{(k)}\right)_{i,:} \end{aligned} \tag{4.43}$$

for some $K \in \mathbb{N}$, where each $\boldsymbol{W}_{(k)} \in \mathbb{R}^{d\times c}$ is a feature transformation, and the matrices $\tilde{\boldsymbol{A}}_{(k)} \in \mathbb{R}^{n\times n}$ contain scalar edge weights satisfying $\left(\tilde{A}_{(k)}\right)_{i,j} = 0$ whenever $\boldsymbol{A}_{i,j} = 0$. The edge-specific feature transformations are defined as $\tilde{\boldsymbol{W}}_{(i,j)} = \sum_{k=1}^{K} \left(\tilde{A}_{(k)}\right)_{i,j} \boldsymbol{W}_{(k)}$ for all $i, j \in [n]$.

The LMGC provides a general framework for message-passing operations that are based on the MIMO-GC. Similar to the MIMO-GC, the LMGC is defined using multiple computational graphs $\tilde{\boldsymbol{A}}_{(1)}, \dots, \tilde{\boldsymbol{A}}_{(K)}$, or equivalently, distinct feature transformations $\tilde{\boldsymbol{W}}_{(i,j)}$ for every edge from node j to node i. These distinct feature transformations are linear combinations of the feature transformations of each computational graph. The coefficients $\left(\tilde{A}_{(k)}\right)_{i,j} \in \mathbb{R}$ can also be interpreted as the K edge weights for the connection from node j to node i. The definition of the LMGC framework leaves the definition of these edge weights and the number of computational graphs open. The feature transformations $\boldsymbol{W}_{(k)}$ are typically learnable parameters. To provide a better understanding of this framework and its generality, we now demonstrate how several existing MPNNs are specific instantiations of the LMGC framework. We start with the GCN (Kipf et al. 2017), as one of the most commonly employed message-passing methods:

Example 4.2 *(GCN (Kipf et al. 2017)).* Let $\boldsymbol{W} \in \mathbb{R}^{d\times c}$ be a feature transformation. The function

$$f_{\text{GCN}}(\boldsymbol{A}, \boldsymbol{X}) = \boldsymbol{A}_{\text{sym}}\boldsymbol{X}\boldsymbol{W} \tag{4.44}$$

is an LMGC with $K = 1$, $\tilde{\boldsymbol{A}}_{(1)} = \boldsymbol{A}_{\text{sym}}$, and $\boldsymbol{W}_{(1)} = \boldsymbol{W}$.

As $K = 1$ for the GCN, this method exhibits SCA as shown in Theorem 3.6 and does not benefit from multiple computational graphs. For other message-passing methods, it is more straightforward to see the equivalence in the node-wise aggregation form. The SAGE convolution used within GraphSAGE (Hamilton et al. 2017) is such an example:

Example 4.3 *(SAGE (Hamilton et al. 2017)).* Let $\boldsymbol{V}_{(1)}, \boldsymbol{V}_{(2)} \in \mathbb{R}^{d\times c}$ be feature transformations. The function

$$(f_{\text{SAGE}}(\boldsymbol{A}, \boldsymbol{X}))_{i,:} = \left(\boldsymbol{X}\boldsymbol{V}_{(1)}\right)_{i,:} + \sum_{j\in N_i} \frac{1}{d_i} \left(\boldsymbol{X}\boldsymbol{V}_{(2)}\right)_{j,:} \tag{4.45}$$

is an LMGC with $K = 2$, $\boldsymbol{W}_{(1)} = \boldsymbol{V}_{(1)}$, $\boldsymbol{W}_{(2)} = \boldsymbol{V}_{(2)}$, and $\left(\tilde{A}_{(1)}\right)_{i,i} = 1$ for every $i \in [n]$ and zero otherwise, and $\left(\tilde{\boldsymbol{A}}_{(2)}\right)_{(i,j)} = \frac{1}{d_i}$ for every $i \in [n]$ and $j \in N_i$ and zero otherwise.

The LMGC is also similar to multi-head attention, as used for the graph attention network (GAT) (Velickovic et al. 2018). The softmax used to compute the attention coefficients constrains them and prevents the resulting matrices $\tilde{\boldsymbol{A}}_{(k)}$ from having independent eigenvectors that can be selectively amplified:

Example 4.4 *(GAT (Velickovic et al. 2018)).* Let $H \in \mathbb{N}$ be the number of attention heads, $\boldsymbol{V}_{(h)} \in \mathbb{R}^{d\times c}$ be feature transformations, and $\alpha_{(h)}^{(i,j)} \in \mathbb{R}$ be attention coefficients for every $h \in [H]$ and edge $i, j \in [n]$ with $\boldsymbol{A}_{i,j} \neq 0$. The function

$$(f_{\text{GAT}}(\boldsymbol{A}, \boldsymbol{X}))_{i,:} = \sum_{h\in[H]} \sum_{j\in N_i} \alpha_{(h)}^{(i,j)} \boldsymbol{X}_{j,:}\boldsymbol{V}_{(h)} \tag{4.46}$$

is an LMGC with $K = H$, $\left(\tilde{A}_{(k)}\right)_{i,j} = \alpha_{(k)}^{(i,j)}$, and $\boldsymbol{W}_{(k)} = \boldsymbol{V}_{(k)}$ for all $k \in [K]$.

Any other linear message-passing step can be expressed as an equivalent LMGC, including more complex ones like the GatedGCN (Dwivedi, Joshi, et al. 2023) or neural sheaf diffusion (Hansen et al. 2020). The LMGC framework allows a more streamlined development of novel MPNNs and the analysis of their theoretical properties, as we only need to consider the effect of different choices used for $\left(\tilde{A}_{(k)}\right)_{i,j}$. We now study properties of LMGCs induced by specific choices of edge weights $\left(\tilde{A}_{(k)}\right)_{i,j}$.

Theoretical Properties of LMGCs

Several of our previous results already apply to specific instantiations of the LMGC. Based on Theorem 3.6, we have shown that any LMGC with $K = 1$ exhibits the SCA property. Following Theorem 4.2 and Theorem 4.1, we have shown that structurally independent nodes can avoid SCA. We will now further study the properties of LMGCs with specific choices for edge weights $\left(\tilde{A}_{(k)}\right)_{i,j}$.

While a linear message-passing step can be expressed as an LMGC, non-linear feature transformations cannot be represented by the framework. However, such transformations are utilized within the graph isomorphism network (GIN) to allow for the injectivity of each message-passing step. This is a key property of message-passing to distinguish the same set of non-isomorphic graphs as the Weisfeiler-Leman (WL) test (Weisfeiler et al. 1968; Xu, Hu, et al. 2019; Morris, Ritzert, et al. 2019). While the GIN cannot be represented as an LMGC, we now show that the LMGC is sufficient for allowing injectivity in message-passing:

Proposition 4.5 (Injectivity). Let $\mathcal{X} \subset \mathbb{R}^d$ be a countable set and $K \in \mathbb{N}_{>0}$. Consider the LMGC function

$$f\left(\boldsymbol{x}_{(i)}, \mathcal{X}_i\right) = \sum_{\boldsymbol{x}_{(j)} \in \mathcal{X}_i} \tilde{\boldsymbol{W}}_{(i,j)} \boldsymbol{x}_{(j)}\,, \tag{4.47}$$

where $\mathcal{X}_i \subset \mathcal{X}$ is a finite multiset and $\tilde{\boldsymbol{W}}_{(i,j)} = \sum_{k=1}^{K} a_{(i,j)}^{(k)} \boldsymbol{W}_{(k)}$ with scalar coefficients $a_{(i,j)}^{(k)} \in \mathbb{R}$ and matrices $\boldsymbol{W}_{(k)} \in \mathbb{R}^{d \times c}$. Then, $f\left(\boldsymbol{x}_{(i)}, \mathcal{X}_i\right)$ is injective with respect to the pair $\left(\boldsymbol{x}_{(i)}, \mathcal{X}_i\right)$ for a.e. choice of coefficients $a_{(i,j)}^{(k)}$ and $\boldsymbol{W}_{(k)}$.

Proof. To prove injectivity, we need to show that

$$f\left(\boldsymbol{x}_{(i)}, \mathcal{X}_i\right) - f\left(\boldsymbol{x}_{(p)}, \mathcal{X}_p\right) \neq 0 \tag{4.48}$$

for any $\boldsymbol{x}_{(i)}, \boldsymbol{x}_{(p)} \in \mathcal{X}$ and $\mathcal{X}_i, \mathcal{X}_p \subset \mathcal{X}$ with $\boldsymbol{x}_{(i)} \neq \boldsymbol{x}_{(p)}$ or $\mathcal{X}_i \neq \mathcal{X}_p$.

Substituting the definition of f, we get

$$\sum_{\boldsymbol{x}_{(j)} \in \mathcal{X}_i} \sum_{k=1}^{K} a_{(i,j)}^{(k)} \boldsymbol{W}_{(k)} \boldsymbol{x}_{(j)} - \sum_{\boldsymbol{x}_{(q)} \in \mathcal{X}_p} \sum_{k=1}^{K} a_{(p,q)}^{(k)} \boldsymbol{W}_{(k)} \boldsymbol{x}_{(q)} \neq 0\,. \tag{4.49}$$

This can be rewritten as a linear combination of the feature transformations $\boldsymbol{W}_{(k)}$:

$$\sum_{k=1}^{K} \boldsymbol{W}_{(k)} \left(\sum_{\boldsymbol{x}_{(j)} \in \mathcal{X}_i} a_{(i,j)}^{(k)} \boldsymbol{x}_{(j)} - \sum_{\boldsymbol{x}_{(q)} \in \mathcal{X}_p} a_{(p,q)}^{(k)} \boldsymbol{x}_{(q)} \right) \neq 0\,. \tag{4.50}$$

Based on the a.e. constraint on the feature transformations $\boldsymbol{W}_{(k)}$, this is zero if and only if the weighted sums are zero simultaneously. As such, we require

$$\sum_{\boldsymbol{x}_{(j)} \in \mathcal{X}_i} a_{(i,j)}^{(k)} \boldsymbol{x}_{(j)} - \sum_{\boldsymbol{x}_{(q)} \in \mathcal{X}_p} a_{(p,q)}^{(k)} \boldsymbol{x}_{(q)} \neq 0 \tag{4.51}$$

for at least one $k \in [K]$. This is zero only on a set of measure zero. Hence, for a.e. choice of coefficients $\{a_{(i,j)}^{(k)}\}$, we have $f(\boldsymbol{x}_{(i)}, \mathcal{X}_i) - f(\boldsymbol{x}_{(p)}, \mathcal{X}_p) \neq 0$. □

This injectivity result is very similar to the injectivity of the GIN (Xu, Hu, et al. 2019). Proposition 4.5 confirms that linear MPNNs are sufficient to allow for injectivity in message-passing. Thus, such an LMGC achieves the same expressive power for distinguishing non-isomorphic graphs as the WL test (Weisfeiler et al. 1968), which is maximal for any MPNN (Xu, Hu, et al. 2019; Morris, Ritzert, et al. 2019). This is already achievable with a single computational graph. Based on the proof, the representation resulting from each computational graph is injective. With multiple computational graphs, we show that their combination yields linearly independent representations. This allows an LMGC to avoid SCA and amplify distinct components across feature channels:

Theorem 4.4 (Linear Independence). Let $\mathcal{X} \subset \mathbb{R}^d$ be a countable set and $K \in \mathbb{N}_{>0}$. Consider the LMGC function

$$f\left(\boldsymbol{x}_{(i)}, \mathcal{X}_i\right) = \sum_{\boldsymbol{x}_{(j)} \in \mathcal{X}_i} \tilde{\boldsymbol{W}}_{(i,j)} \boldsymbol{x}_{(j)}\,, \tag{4.52}$$

where $\mathcal{X}_i \subset \mathcal{X}$ is a finite multiset and $\tilde{\boldsymbol{W}}_{(i,j)} = \sum_{k=1}^{K} a_{(i,j)}^{(k)} \boldsymbol{W}_{(k)}$ with scalar coefficients $a_{(i,j)}^{(k)} \in \mathbb{R}$ and matrices $\boldsymbol{W}_{(k)} \in \mathbb{R}^{d \times c}$. Suppose $(\boldsymbol{x}_{(i)}, \mathcal{X}_i)$ and $(\boldsymbol{x}_{(p)}, \mathcal{X}_p)$ satisfy $\boldsymbol{x}_{(i)} \neq \boldsymbol{x}_{(p)}$ or $\mathcal{X}_i \neq m \cdot \mathcal{X}_p$ for any $m \in \mathbb{N}$ Then, $f\left(\boldsymbol{x}_{(i)}, \mathcal{X}_i\right)$ and $f\left(\boldsymbol{x}_{(p)}, \mathcal{X}_p\right)$ are linearly independent for a.e. choice of coefficients $a_{(i,j)}^{(k)}$ and $\boldsymbol{W}^{(k)}$.

Proof. To prove linear independence, we need to show that

$$f\left(\boldsymbol{x}_{(i)}, \mathcal{X}_i\right) - b \cdot f\left(\boldsymbol{x}_{(p)}, \mathcal{X}_p\right) \neq 0 \tag{4.53}$$

for any $b \in \mathbb{R}$. Following the proof of Proposition 4.5, this is equal to

$$\sum_{k=1}^{K} \boldsymbol{W}_{(k)} \left(\sum_{\boldsymbol{x}_{(j)} \in \mathcal{X}_i} a_{(i,j)}^{(k)} \boldsymbol{x}_{(j)} - b \cdot \sum_{\boldsymbol{x}_{(q)} \in \mathcal{X}_p} a_{(p,q)}^{(k)} \boldsymbol{x}_{(q)} \right) \neq 0\,. \tag{4.54}$$

This is satisfied for a.e. $\boldsymbol{W}_{(k)}$ if and only if

$$\sum_{\boldsymbol{x}_{(j)} \in \mathcal{X}_i} a_{(i,j)}^{(k)} \boldsymbol{x}_{(j)} - b \cdot \sum_{\boldsymbol{x}_{(q)} \in \mathcal{X}_p} a_{(p,q)}^{(k)} \boldsymbol{x}_{(q)} \neq 0\,. \tag{4.55}$$

for at least one $k \in [K]$. For the case that not all $\boldsymbol{x}_{(j)} \in \mathcal{X}_i$ and $\boldsymbol{x}_{(q)} \in \mathcal{X}_p$ are linearly dependent, this holds for all $k \in [K]$ for a.e. coefficients $a_{(i,j)}^{(k)}$. For the case that all $\boldsymbol{x}_{(j)} \in \mathcal{X}_i$ and $\boldsymbol{x}_{(q)}\mathcal{X}_p$ are linearly dependent and $K = 1$, $(\boldsymbol{x}_{(i)}, \mathcal{X}_i)$ and $f\left(\boldsymbol{x}_{(p)}, \mathcal{X}_p\right)$ will be linearly dependent, and b can be chosen accordingly. However, each computational graph requires a different b for a.e. choice of coefficients. Hence, for $K > 1$ we have $f\left(\boldsymbol{x}_{(i)}, \mathcal{X}_i\right) - b \cdot f\left(\boldsymbol{x}_{(p)}, \mathcal{X}_p\right) \neq 0$ for any $b \in \mathbb{R}$. □

This finding shows that LMGCs avoid SCA for a.e. choice of feature transformations and edge weights for $K > 1$. This enables the LMGC to yield more similar adjacent node representations across some of the feature channels, while simultaneously allowing for less similar features in other channels. This connection also highlights the close connections between improving the expressive power of an MPNN and solving the underlying phenomena behind over-smoothing and rank collapse. It also highlights the connection between SCA and the structural expressivity of the WL test. As linear independence subsumes injectivity, avoiding SCA and over-smoothing with such an LMGC also results in a method that is maximally expressive for message-passing methods in distinguishing non-isomorphic graphs. While the linear independence requires $\mathcal{X}_i \neq m \cdot \mathcal{X}_p$ for any m, injectivity is still ensured for this case. Thus, in the second iteration, we will have $\boldsymbol{x}_{(i)} \neq \boldsymbol{x}_{(p)}$, resulting in linearly independent node representations. Linear independence further improves the optimization process, as it allows the feature transformations to control the amplification of components for each feature channel individually.

Each $a_{(k)}^{(i,j)}$ in Proposition 4.5 and Theorem 4.4 can be independently obtained, e.g., by a function $a_{(k)}^{(i,j)} = \phi_k\left(\boldsymbol{x}_{(i)}, \boldsymbol{x}_{(j)}\right) \in \mathbb{R}$ of the node representations $\boldsymbol{x}_{(i)}$ and $\boldsymbol{x}_{(j)}$. Given the measure-zero set for coefficients $a_{(k)}^{(i,j)}$ for which injectivity and linear independence are not given, many functions ϕ_k can be defined that avoid this measure-zero set for the edge weights as their output. Neural networks and MLPs specifically can then be designed to represent or approximate such functions. However, the methods presented as examples above do not satisfy this condition. Due to the softmax applied within attention-based methods like GAT (Velickovic et al. 2018) and GATv2(Brody et al. 2022), the coefficients form a measure-zero set. The averaging prevents such methods from distinguishing multisets with different multiplicities (Xu, Hu, et al. 2019), e.g., $\mathcal{X}_{(p)} = \{\{\boldsymbol{x}\}\}$ and $\mathcal{X}_{(q)} = \{\{\boldsymbol{x}, \boldsymbol{x}\}\}$ for any $\boldsymbol{x} \in \mathcal{X}$. Other methods were proposed that utilize the tanh activation function for coefficients, which generally allows such methods to avoid measure-zero subspaces.

Further properties of LMGCs can be similarly studied by considering the effect of coefficients $\left(\tilde{A}_{(k)}\right)_{(i,j)}$. To provide an example that follows the definition of the LMGC and can satisfy Proposition 4.5 and Theorem 4.4, we introduce the following instantiation of the LMGC framework.

Table 4.1: Results of the MRS framework applied to the GCN and the ZINC12k dataset. Training and testing mean absolute error (MAE) are independently obtained using separate hyperparameter optimizations for the learning rate and number of iterations. Step time refers to the runtime for one optimization step in milliseconds (ms). Average results over three random seeds are reported.

Method	Step Time (ms)	ZINC12k (MAE) Training	Testing
GCN	$\mathbf{4.3} \pm 0.1$	0.051 ± 0.002	0.404 ± 0.011
MRS-GCN$_{\text{DA}}$ (random)	5.7 ± 0.1	0.006 ± 0.001	0.623 ± 0.003
MRS-GCN$_{\text{DA}}$ (Features)	5.8 ± 0.4	0.011 ± 0.003	0.390 ± 0.004
MRS-GCN$_{\text{DA}}$ (PPR)	6.8 ± 0.4	0.010 ± 0.002	0.358 ± 0.006
MRS-GCN$_{\text{DA}}$ (Degree)	5.8 ± 0.2	$\mathbf{0.003} \pm 0.001$	$\mathbf{0.318} \pm 0.031$

Defining an LMGC

Based on the definition of LMGCs in Definition 4.4, we only need to select the number of computational graphs K and the method to obtain edge weights $\left(\tilde{A}_{(k)}\right)_{(i,j)}$ for all $k \in [K]$ and all edges. Following Proposition 4.5 and Theorem 4.4, we apply a neural network based on the node representations of the adjacent nodes for each edge. We define the coefficients similar to GATv2 while applying a tanh activation function as the last operation, as in FAGCN (Bo et al. 2021) and GGCN (Yan et al. 2022):

$$\left(\tilde{A}_{(k)}\right)_{(i,j)} := \sigma_2 \left(\sigma_1 \left(\boldsymbol{X}_{i,:}\boldsymbol{W}_{(1)} || \dots || \boldsymbol{X}_{i,:}\boldsymbol{W}_{(K)} || \boldsymbol{X}_{j,:}\boldsymbol{W}_{(1)} || \dots || \boldsymbol{X}_{j,:}\boldsymbol{W}_{(K)}\right) \boldsymbol{v}_{(k)}\right) \tag{4.56}$$

where $||$ is the concatention operator, σ_1 is the LeakyReLU activation function, σ_2 is the tanh activation function, $\boldsymbol{W}_{(1)}, \dots, \boldsymbol{W}_{(K)} \in \mathbb{R}^{d \times c}$ are the feature transformations of the LMGC, and $\boldsymbol{X}_{i,:}, \boldsymbol{X}_{j,:} \in \mathbb{R}^{1 \times d}$ are the i-th and j-th row of $\boldsymbol{X}$, respectively, and $\boldsymbol{v}_{(k)} \in \mathbb{R}^{2 \cdot K \cdot c}$ is a vector of learnable parameters that is distinct per computational graph. While this construction of the edge weights is designed as a combination of previous methods, future work can further study which functions ϕ_k are best suited to satisfy the properties required in Proposition 4.5 and Theorem 4.4.

4.5 Evaluation

We now evaluate the MRS framework and the LMGC framework, both of which propose multiple computational graphs but follow different derivations. In Section 4.5.1, we apply the MRS framework to established MPNNs and compare their empirical performance. In Section 4.5.2, we compare the performance of the LMGC derived from the MIMO-GC with several existing MPNNs.

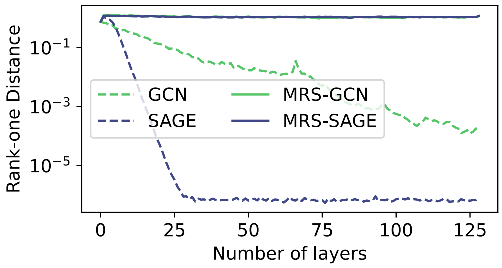


Figure 4.5: Rank-one distance (ROD) on Cora (McCallum et al. 2000) after up to 128 iterations of the GCN, SAGE, and their respective MRS versions. Average values over 50 random parameter initializations are displayed.

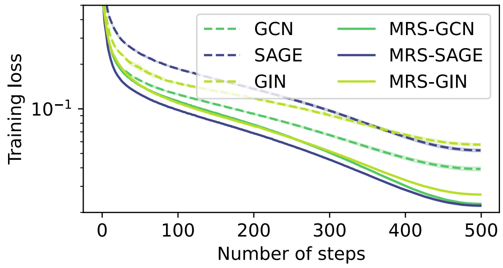


Figure 4.6: Training error during optimization when applying the MRS framework for the ZINC dataset using the methods GCN, SAGE, GIN, and their respective MRS versions. The learning rate and the number of iterations are tuned for each method. Mean values over three random initializations.

4.5.1 The MRS Framework

Starting with the evaluation of the MRS framework, we empirically study the effect of applying this method to several MPNNs across various tasks. Specifically, we study the benefits of MRS on optimization and then use it on a set of benchmark tasks. This set of experiments is based on Roth, Bause, et al. (2024). Our reproducible code is publicly available [3] .

More Effective Parameters

Based on our insights from Section 4.3.1, the MRS framework enables more effective feature transformations and learnable parameters, resulting in improved optimization. We conduct several experiments and ablations using the ZINC dataset (Sterling et al. 2015). It consists of around 250 000 graphs, for which each node corresponds to an atom and the edges describe bonds between them. ZINC is a graph-level regression task, where the task is to predict the constrained solubility of each molecule represented as a graph. In addition, we use ZINC12k, a subset consisting of 12 000 graphs as a computationally less consuming alternative, as proposed by Dwivedi, Joshi, et al. (2023). We use the experimental setup and implementation given by Dwivedi, Rampásek, et al. (2022) and the improvements proposed by Tönshoff et al. (2024). Following their experimental setup, each method utilizes at most 500 000 learnable parameters, and AdamW (Loshchilov et al. 2019) is employed as the optimizer. All experiments are repeated three times using random seeds, and the average and standard deviation of the results are presented.

[3] https://github.com/roth-andreas/splitting-computational-graphs

Table 4.2: Results on ZINC for applying the MRS framework to the five methods GCN (Kipf et al. 2017), SAGE (Hamilton et al. 2017), GAT (Velickovic et al. 2018), GIN (Xu, Hu, et al. 2019), GatedGCN (Dwivedi, Joshi, et al. 2023). Training and testing are obtained using independent hyperparameter optimization of the number of iterations and the learning rate. Results are averaged over three random initializations. Pairwise best results in **bold**.

Numer of iterations	ZINC (MAE)	
	Training	Testing
GCN	0.053 ± 0.001	0.155 ± 0.003
MRS-GCN$_{\text{DA}}$	$\mathbf{0.023} \pm 0.000$	$\mathbf{0.134} \pm 0.001$
SAGE	0.039 ± 0.001	0.123 ± 0.002
MRS-SAGE$_{\text{DA}}$	$\mathbf{0.022} \pm 0.000$	$\mathbf{0.106} \pm 0.002$
GAT	0.049 ± 0.002	0.149 ± 0.001
MRS-GAT$_{\text{DA}}$	$\mathbf{0.026} \pm 0.001$	$\mathbf{0.128} \pm 0.000$
GIN	0.058 ± 0.001	0.123 ± 0.004
MRS-GIN$_{\text{DA}}$	$\mathbf{0.026} \pm 0.000$	$\mathbf{0.106} \pm 0.004$
GatedGCN	0.051 ± 0.002	0.099 ± 0.011
MRS-GatedGCN$_{\text{DA}}$	$\mathbf{0.011} \pm 0.003$	$\mathbf{0.088} \pm 0.004$

The Effect of Different Partial Orderings Following Section 4.3.2, we convert an MPNN into an MRS-MPNN by defining a strict partial ordering, leading to three directed acyclic relations. Here, we consider four different strict partial orderings and study their effectiveness. To show that any ordering improves the optimization, we first employ a random ordering. However, messages exchanged for each relation do not exhibit any similarity, which can hurt generalization. Second, we evaluate a feature-based approach where all initial features are summed. Third, we use personalized PageRank (Page et al. 1999) as a node centrality measure. Lastly, we use the node degree as a simple and efficient ordering. We apply and compare all splits to the GCN, referring to our methods as MRS-GCN$_{\text{DA}}$. While these are relatively simple orderings, we aim to show the benefit of splitting the edges in general. For each method, the learning rate and the number of layers are tuned.

Results are presented in Table 4.1. We observe that all MRS versions achieve a significantly reduced training loss, by up to 94% for the node degree-based splitting. Even the random split achieves a similarly low training loss. This provides further evidence that any split leads to improvements in optimization. The generalization is also improved for all methods, except the random split. As there are no similarities between the edges in a random relation, the generalization performance is not improved. The node degree also achieves the lowest testing loss, reducing the error by 21% compared to the GCN. The runtime is slightly higher for all MRS methods, with the degree MRS resulting in 5.8 milliseconds, compared to 4.3 milliseconds for each optimization step of the GCN. Based on

Table 4.3: Ablation study on ZINC12k for combining the MRS framework with residual connections (+ Res), jumping knowledge (Xu, C. Li, et al. 2018) (+ JK), and Laplacian positional encoding (Dwivedi, Joshi, et al. 2023) (+ LapPE). The number of iterations indicates the number of message-passing iterations applied. The learning rate is tuned for each entry independently. The mean average error (MAE) over three random initializations is reported. Pairwise best results in **bold**.

Method	1	2	4	8	16	32
GCN	0.591	0.493	0.421	0.404	0.417	0.440
MRS-GCN_{DA}	**0.525**	**0.445**	**0.343**	**0.318**	**0.318**	**0.338**
GCN + Res	0.567	0.471	0.427	0.403	0.370	0.336
MRS-GCN_{DA} + Res	**0.508**	**0.402**	**0.295**	**0.257**	**0.252**	**0.250**
GCN + JK	0.588	0.496	0.424	0.409	0.413	0.435
MRS-GCN_{DA} + JK	**0.526**	**0.442**	**0.324**	**0.303**	**0.311**	**0.304**
GCN + LapPE	0.498	0.437	0.392	0.367	0.383	0.444
MRS-GCN_{DA} + LapPE	**0.441**	**0.363**	**0.292**	**0.272**	**0.297**	**0.317**

these initial results, we will use the node degree as a strict partial ordering for all other experiments.

Preventing Rank Collapse As the goal of employing multiple computational graphs was to prevent rank collapse by dealing with SCA as one of its underlying phenomena, we now evaluate the ability of the MRS framework to prevent rank collapse. Following Definition 3.6 of the rank-one distance (ROD) as a metric to quantify the distance of representations to any rank-one matrix, we now measure ROD for the GCN and SAGE as two methods known to exhibit rank collapse and their respective MRS versions. Using a random graph from the ZINC dataset, we apply the respective message-passing methods interleaved with a ReLU activation function for 128 iterations. We compute ROD after each iteration. We present average results across 50 graphs in Figure 4.5. While the ROD decreases quickly for the GCN and SAGE, it remains constant for MRS-GCN and MRS-SAGE across all iterations. This provides further evidence that preventing SCA also deals with rank collapse and over-smoothing.

MRS Applied to Various MPNNs To gain a better understanding of the impact of the MRS framework on message-passing, we now present results where we applied MRS to various established MPNNs. We consider the GCN (Kipf et al. 2017), SAGE (Hamilton et al. 2017), GAT (Velickovic et al. 2018), GIN (Xu, Hu, et al. 2019), and GatedGCN (Dwivedi, Joshi, et al. 2023). For all methods, we optimize the learning rate and number of message-passing iterations. We present results for these methods and their respective MRS versions in Table 4.2. The lowest achieved training loss is significantly reduced for all MRS versions, confirming the benefits of optimizing our MRS-MPNNs. The testing loss is also

Table 4.4: Results on applying the MRS framework to the Dir-GNN (Rossi et al. 2023) for five directed heterophilic benchmark graphs. Equivalent to the Dir-GNN, SAGE is used for Roman-Empire, while the GCN is used for all other tasks. Reported values are the mean and standard deviation over five runs. Best results in **bold**, second-best underlined.

Method	Squirrel	Chameleon	Arxiv-year	Snap-patents	Roman-Empire
MLP	28.77 ± 1.56	46.21 ± 2.99	36.70 ± 0.21	31.34 ± 0.05	64.94 ± 0.62
GCN	53.43 ± 2.01	64.82 ± 2.24	46.02 ± 0.26	51.02 ± 0.06	73.69 ± 0.74
H_2GCN	37.90 ± 2.02	59.39 ± 1.98	49.09 ± 0.10	OOM	60.11 ± 0.52
GPR-GNN	54.35 ± 0.87	62.85 ± 2.90	45.07 ± 0.21	40.19 ± 0.03	64.85 ± 0.27
LINKX	61.81 ± 1.80	68.42 ± 1.38	56.00 ± 0.17	61.95 ± 0.12	37.55 ± 0.36
FSGNN	74.10 ± 1.89	78.27 ± 1.28	50.47 ± 0.21	65.07 ± 0.03	79.92 ± 0.56
ACM-GCN	67.40 ± 2.21	74.76 ± 2.20	47.37 ± 0.59	55.14 ± 0.16	69.66 ± 0.62
GloGNN	57.88 ± 1.76	71.21 ± 1.84	54.79 ± 0.25	62.09 ± 0.27	59.63 ± 0.69
Grad. Gating	64.26 ± 2.38	71.40 ± 2.38	63.30 ± 1.84	69.50 ± 0.39	82.16 ± 0.78
DiGCN	37.74 ± 1.54	52.24 ± 3.65	OOM	OOM	52.71 ± 0.32
MagNet	39.01 ± 1.93	58.22 ± 2.87	60.29 ± 0.27	OOM	88.07 ± 0.27
Dir-GNN	$\underline{75.31} \pm 1.92$	$\underline{79.71} \pm 1.26$	$\underline{64.08} \pm 0.30$	$\underline{73.95} \pm 0.05$	$\underline{91.23} \pm 0.32$
MRS-Dir-GNN$_{\text{DA}}$	$\mathbf{76.01} \pm 1.90$	$\mathbf{80.17} \pm 1.88$	$\mathbf{66.03} \pm 0.20$	$\mathbf{74.72} \pm 0.05$	$\mathbf{91.87} \pm 0.42$

improved in all cases, though not as significantly. This indicates that the identified optimization benefits can also lead to improvements in generalization. In Figure 4.6, we also visualize the training loss during optimization for the GCN, SAGE, GIN, and their respective MRS versions. It confirms that optimization is improved, as the loss is reduced more quickly, leading it to converge to a lower final value.

In addition to these message-passing methods, MPNNs typically combine these methods with other approaches like residual connections (K. He et al. 2016; M. Chen et al. 2020), jumping knowledge (Xu, C. Li, et al. 2018), or Laplacian positional encoding (Dwivedi, Joshi, et al. 2023). We present results for combining these methods with the GCN and the MRS-GCN in Table 4.3, which includes detailed results for various numbers of iterations in $\{1, 2, 4, 8, 16, 32\}$. In all cases, the MRS-GCN achieves better results. Crucially, as studied with SCA, improvements are already significant after a single message-passing iteration. The best result for the MRS-GCN is achieved in combination with residual connections and 32 message-passing iterations. While for the other methods, 32 iterations do not achieve the lowest testing error, this suggests potential for future work to investigate vanishing gradient issues.

The MRS framework and State-of-the-art methods

With these promising results, we now also apply the MRS framework to state-of-the-art methods for complex tasks. We consider the set of large-scale directed heterophilic benchmark graphs based on the implementation of Rossi et al. (2023). These datasets are the Squirrel and Chameleon graphs (Pei et al. 2020), the arxiv-year and snap-patents graphs (Lim et al. 2021), and the roman-empire graph (Platonov et al. 2023). With Dir-

Method	MSE
GATv2	0.12 ± 0.04
FAGCN	0.68 ± 0.02
ACM	0.49 ± 0.02
GIN	$\underline{0.08} \pm 0.03$
LMGC	$\mathbf{25 \cdot 10^{-9}} \pm 86 \cdot 10^{-11}$

Table 4.5: Minimal training mean squared error (MSE) for the universality task. Given random representations $\boldsymbol{X}$ and $\boldsymbol{X}'$ for a random ER-graph, a single iteration of each of the considered methods is applied to $\boldsymbol{X}$. The MSE between the output and $\boldsymbol{X}'$ is minimized for 4000 optimization steps. The average and standard deviation are shown over three random seeds. The best result is in **bold**, the second-best underlined.

	ZINC		
Method	Train	Test	Time per Epoch (s)
GATv2	0.077 ± 0.001	0.114 ± 0.004	52.7 ± 0.0
FAGCN	0.093 ± 0.008	0.130 ± 0.003	52.9 ± 1.1
ACM	0.109 ± 0.003	0.128 ± 0.001	73.1 ± 0.9
GIN	$\underline{0.068} \pm 0.001$	$\underline{0.088} \pm 0.002$	$\mathbf{35.5} \pm 1.0$
LMGC	$\mathbf{0.054} \pm 0.001$	$\mathbf{0.080} \pm 0.001$	$\underline{47.5} \pm 0.5$

Table 4.6: Mean absolute error (MAE) results on the ZINC dataset. Each method uses at most $100\,000$ learnable parameters. Training and testing MAE are obtained using independent hyperparameter optimizations. Time per epoch is in seconds (s). Best results in **bold**, second-best underlined.

GNN (Rossi et al. 2023) having achieved state-of-the-art results for these datasets by using the graph with reverse edges as a second computational graph, we now apply the MRS framework to this method to study its effectiveness for more complex methods. Dir-GNN applies Dir-GCN to squirrel, chameleon, arxiv-year, and snap-patents, and Dir-SAGE to roman-empire. We use the equivalent MRS version for each of these datasets. Following their experimental setup, we use ten fixed splits for squirrel, chameleon, and roman-empire, and five fixed splits for arxiv-year and snap-patents. Following their hyperparameters, we also optimize the learning rate, number of iterations, and dropout ratio using a grid search. We also provide baseline comparisons for ten state-of-the-art methods.

We present average results in Table 4.4. The MRS-Dir-GNN consistently improves results for all tasks. We observe that the differences in results are more significant for the two larger graphs (arxiv-year and snap-patents), aligning with our previous findings that more complex functions can be represented. For squirrel, chameleon, and roman-empire, the other methods already achieve a low training error, leaving less potential for improved capabilities.

	ZINC12k		
Method	Train	Test	Time per Epoch (s)
GATv2	$\underline{0.005} \pm 0.003$	0.377 ± 0.024	2.5 ± 0.1
FAGCN	0.016 ± 0.006	0.365 ± 0.018	2.5 ± 0.0
ACM	0.019 ± 0.005	0.278 ± 0.006	3.4 ± 0.0
GIN	0.018 ± 0.009	$\underline{0.272} \pm 0.009$	$\mathbf{1.7} \pm 0.0$
LMGC	$\mathbf{0.002} \pm 0.001$	$\mathbf{0.241} \pm 0.018$	$\underline{2.4} \pm 0.3$

Table 4.7: Mean absolute error (MAE) results on the ZINC12k dataset. Each method uses at most 100 000 learnable parameters. Training and testing MAE are obtained using independent hyperparameter optimizations. Time per epoch is in seconds (s). Best results in **bold**, second-best underlined.

Method	Basic	+ LapPE	+ Jumping Knowledge	+ Residual	+ All three
GATv2	0.377 ± 0.024	0.341 ± 0.040	0.388 ± 0.017	0.311 ± 0.016	0.294 ± 0.019
FAGCN	0.365 ± 0.018	0.349 ± 0.038	0.352 ± 0.042	0.289 ± 0.019	0.232 ± 0.012
ACM	0.278 ± 0.006	0.281 ± 0.019	0.288 ± 0.008	0.266 ± 0.017	0.238 ± 0.006
GIN	$\underline{0.272} \pm 0.009$	$\underline{0.259} \pm 0.012$	$\underline{0.267} \pm 0.020$	$\underline{0.240} \pm 0.005$	$\underline{0.228} \pm 0.014$
LMGC	$\mathbf{0.241} \pm 0.018$	$\mathbf{0.234} \pm 0.009$	$\mathbf{0.233} \pm 0.019$	$\mathbf{0.215} \pm 0.006$	$\mathbf{0.203} \pm 0.004$

Table 4.8: Mean absolute error (MAE) results on the ZINC12k dataset. Each method is combined with Laplacian positional encoding (+ LapPE) (Dwivedi, Joshi, et al. 2023), jumping knowledge (Xu, C. Li, et al. 2018), residual connections (K. He et al. 2016), and a combination of these three modifications. At most 100 000 learnable parameters are used. Each result is obtained using an independent hyperparameter optimization. Best results in **bold**, second-best underlined.

4.5.2 The LMGC Framework

As the second approach to obtain multiple computational graphs, we now evaluate the benefits of the LMGC framework. As described in Proposition 4.5, the LMGC can be constructed to be injective on multisets, as GIN (Xu, Hu, et al. 2019), while also being able to adaptively filter incoming messages similar to the attention-based message-passing methods (Velickovic et al. 2018; Brody et al. 2022). At the same time, the LMGC also allows for negative edge weights as in FAGCN (Bo et al. 2021), which can act as a high-pass filter. The LMGC also prevents SCA as shown in Theorem 4.4, leading to more informative node representations and improved optimization. We consider graph-level and node-level tasks to confirm these properties. These experiments are based on Roth and Liebig (2025). Our reproducible implementation is publicly available [4] .

[4] `https://github.com/roth-andreas/mimo-graph-convolutions`

Method	Texas	Cornell	Wisconsin	Film	Chameleon	Squirrel
GATv2	71.6 ± 1.0	66.1 ± 0.6	79.1 ± 2.0	35.1 ± 0.2	$\underline{47.1} \pm 0.3$	$\underline{35.1} \pm 0.2$
FAGCN	$\underline{73.5} \pm 1.8$	$\underline{68.1} \pm 1.9$	$\underline{80.2} \pm 1.8$	$\underline{36.0} \pm 0.3$	46.9 ± 0.5	34.6 ± 0.3
ACM	72.3 ± 0.4	65.1 ± 0.7	74.2 ± 0.9	35.8 ± 0.3	45.5 ± 0.9	34.5 ± 0.1
GIN	70.5 ± 1.1	66.1 ± 1.0	79.0 ± 0.6	34.1 ± 0.3	46.1 ± 0.4	34.6 ± 0.5
LMGC	$\mathbf{74.2} \pm 2.2$	$\mathbf{68.9} \pm 2.2$	$\mathbf{81.4} \pm 1.1$	$\mathbf{36.3} \pm 0.4$	$\mathbf{49.8} \pm 0.8$	$\mathbf{35.9} \pm 0.5$

Table 4.9: Obtained testing accuracies on six heterophilic node classification tasks. An independent hyperparameter optimization is performed for each entry. Each method uses at most 100 000 learnable parameters. Best results in **bold**, second-best underlined.

Evaluated Methods

Based on our identified benefits of the LMGC, we compare it with the following four established and related methods.

GIN GIN (Xu, Hu, et al. 2019) ensures injectivity on multisets by applying a multi-layer perceptron on the multisets of node representations. GIN is maximally expressive for MPNNs and equivalent to the expressive power of the WL test. While this property is similar to Proposition 4.5, GIN cannot be represented by an LMGC due to its non-linear feature transformation. We evaluate a standard version of the GIN that utilizes a two-layer MLP with ReLU activations after aggregation.

GATv2 As one effective attention-based message-passing operation, GATv2 (Brody et al. 2022) uses learnable attention coefficients as edge weights. With these, GATv2 can weight the importance of incoming messages adaptively. It utilizes H attention heads. It is equivalent to an LMGC with $K = H$ computational graphs and coefficients

$$\left(\tilde{A}_{(k)}\right)_{i,j} = \sigma_2 \left(\sigma_1 \left(\boldsymbol{X}_{i,:}\boldsymbol{W}_{(k)} + \boldsymbol{X}_{j,:}\boldsymbol{W}_{(k)}\right) \boldsymbol{v}_{(k)}\right) \tag{4.57}$$

where $\boldsymbol{W}_{(k)} \in \mathbb{R}^{d \times c}$ are the feature transformations, $\boldsymbol{v}_{(k)} \in \mathbb{R}^{c}$ is a vector, σ_1 is the LeakyReLU activation function and σ_2 is a node and head-wise softmax activation. To ensure fairness, we employ $K = 4$ computational graphs for GATv2 and our LMGC instantiation.

ACM As one method that has been proposed to utilize multiple computational graphs, we evaluate the adaptive-channel mixing (ACM) method (Luan et al. 2022). They propose to use a normalized adjacency matrix to amplify low-frequency components, a normalized graph Laplacian to amplify high-frequency components, and a residual connection as three computational graphs. Expressed in the LMGC framework this corresponds to setting $K = 3$ and selecting the matrices of coefficients as $\boldsymbol{A}^{(1)} = \boldsymbol{A}_{\text{sym}}$, $\boldsymbol{A}^{(2)} = \boldsymbol{L}_{\text{sym}}$ and $\boldsymbol{A}^{(3)} = \boldsymbol{I}_n$. To ensure fairness, we only use the third computational graph when we use residual connections for all methods.

FAGCN The frequency adaptation GCN (FAGCN) (Bo et al. 2021) allows for negative edge weights, which can act as a high-pass filter. Nodes with a negative edge weight can result in more distant representations after aggregation. This method was specifically designed for heterophilic graphs, where high-frequency components are beneficial. Stated in the LMGC framework, this corresponds to setting $K = 1$ and defining the coefficients as

$$\tilde{A}^{(1)}_{(i,j)} = \frac{\sigma\left([\boldsymbol{X}_{i,:}||\boldsymbol{X}_{j,:}]\boldsymbol{v}\right)}{\sqrt{d_i}\sqrt{d_j}} \tag{4.58}$$

where $\boldsymbol{v} \in \mathbb{R}^{2d}$ is a vector of learnable parameters, σ is the tanh activation function and d_i, d_j are the degrees of nodes i and j, respectively.

Universality

Based on Proposition 4.3, the MIMO-GC can represent any function that maps almost any node representation matrix $\boldsymbol{X}$ to any other node representation matrix $\boldsymbol{X}'$. To evaluate the ability of our considered MPNNs to represent such a mapping, we empirically evaluate their capacity to fit such a function. Given a random Erdős–Rényi graph (Erdős et al. 1959) with n nodes, we sample $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ and $\boldsymbol{X}' \in \mathbb{R}^{n\times d}$ element-wise from a standard normal distribution. We apply a single message-passing iteration, and minimize the mean-squared error (MSE) between the output and $\boldsymbol{X}'$ for 4000 optimization steps using the Adam optimizer. We consider $n = 16$ nodes, an edge probability of $p = 10\%$, a feature dimension of $d = 16$, and tune the learning rate in $\{0.03, 0.01, 0.003\}$.

We present average results over three random seeds in Table 4.5. While GIN achieves the lowest optimization error of the four baselines with 0.08, the LMGC achieves a significantly reduced error of $25 \cdot 10^{-9}$. This provides further evidence of the optimization benefits of multiple computational graphs.

Graph Regression

Now, we evaluate the considered methods for graph-level tasks. As in Section 4.5.1, we consider the ZINC (Sterling et al. 2015) and ZINC12k dataset as a subset of $12\,000$ graphs (Dwivedi, Joshi, et al. 2023). For these tasks, the maximal expressivity of GIN allows it to achieve strong results. Based on Proposition 4.5, the LMGC is theoretically able to match this expressive power. As in Section 4.5.1, we use the implementation of Tönshoff et al. (2024) based on the Long Range Graph Benchmark (Dwivedi, Rampásek, et al. 2022). For all five methods, we perform the same hyperparameter optimization on the number of iterations in $\{6, 8, 10\}$ and the learning rate in $\{0.001, 0.0003, 0.0001\}$ as proposed by Tönshoff et al. (2024). Each combination is repeated for four random seeds. Each model uses at most $100\,000$ parameters.

Results for ZINC and ZINC12k are presented in Table 4.6 and Table 4.7, respectively. The LMGC achieves the best results for training and testing sets, with the GIN achieving roughly 10% lower scores, and all other methods significantly worse due to their lower expressivity. An epoch for ZINC takes GIN 35.5 seconds while the LMGC takes 47.5 seconds, which is slightly faster compared to GATv2.

We present an additional ablation study for ZINC12k in Table 4.8. We combine all methods with Laplacian positional encoding (Kreuzer et al. 2021), jumping knowledge (Xu, C. Li, et al. 2018), and residual connections (K. He et al. 2016), and a combination of all three of these methods. The LMGC also achieves the lowest testing errors for all these combinations. These results provide further evidence that the LMGC can match the expressivity of GIN.

Node Classification

For node-level tasks, the ability of attention-based methods to filter information and the ability of the FAGCN and ACM to amplify high-frequency components are essential. We consider the six heterophilic graph benchmark datasets Texas, Cornell, Wisconsin, Film, Chameleon, and Squirrel, with ten data splits, as proposed by Pei et al. (2020). We integrate our considered methods into the implementation provided by Rusch, B. P. Chamberlain, et al. (2023). Each model uses at most $100\,000$ parameters. The learning rate is tuned in $\{0.01, 0.003, 0.001\}$ and dropout ratio in $\{0.0, 0.25, 0.5\}$ using a grid search. For the selected hyperparameters for each method, all ten splits are rerun five times, and the average results are presented.

These average results are presented in Table 4.9. While the LMGC achieves the best results for all considered tasks, the differences here are less significant. These datasets are challenging due to their heterophilic nature but are relatively small-scale. Due to the beneficial optimization properties of the LMGC, results for larger datasets are expected to improve even further.

5 Preventing Component Dominance Based on Personalized PageRank

Many message-passing neural networks (MPNNs) tend to achieve worse performance across various tasks when the number of message-passing iterations is increased. We identified component dominance (CD) as one of multiple related phenomena, referring to a single component of the initial representations getting dominantly amplified. This leads to a loss of all other information, and node representations become increasingly independent of the initial features. In this chapter, we establish the connection between CD and the PageRank algorithm. As personalized PageRank was proposed as a solution for a similar shortcoming of PageRank, we adapt these modifications for MPNNs. This results in the personalized PageRank graph neural network (PPRGNN), a method that allows for infinitely many message-passing iterations, while maintaining information from the initial representations. We propose an efficient method for performing a fixed-point iteration with an absolute stopping criterion to obtain the solution. This approach further improves our understanding of the CD phenomenon in MPNNs and provides an effective solution for it.

5.1 Introduction

When processing graph-structured data, relevant information often needs to be combined from distant, non-adjacent nodes (Stachenfeld et al. 2021; Rampásek and Wolf 2021; Alon et al. 2021; Freitas et al. 2021; Dwivedi, Rampásek, et al. 2022). Message-passing neural networks (MPNNs), a widely used framework for such tasks, iteratively aggregate the representations of the neighbors of each node. Due to their localized nature, l message-passing iterations capture information only from the l-hop neighborhood of each node. As such, multiple iterations need to be performed to combine distant information. Additionally, non-linear activation functions are typically applied only after each message-passing iteration. Therefore, more iterations allow the extraction of more complex node representations. At the same time, the resulting representations should preserve information about the local neighborhood of each node and its initial features.

However, the performance of many MPNNs has been found to degrade when more message-passing iterations are applied. In the literature, over-smoothing is frequently mentioned as a phenomenon related to this performance degradation (Q. Li et al. 2018; Oono et al. 2020). Our theoretical study in Chapter 3 revealed that many MPNNs exhibit shared component amplification (SCA) (Definition 3.1) and component dominance (CD)

(Definition 3.2), two properties that underlie over-smoothing and rank collapse. In Chapter 4, we extensively studied the design of MPNNs that avoid SCA and demonstrated their effectiveness in representing more complex functions.

In this chapter, we focus on designing methods that mitigate CD in MPNNs. Specifically, we relate CD to the PageRank algorithm (Page et al. 1999), where a similar phenomenon has been identified. This connection to a well-established algorithm with rich theoretical foundations provides a better understanding and intuition for CD in MPNNs. It allows the design of novel MPNNs inspired by modifications proposed for PageRank. To address this issue in PageRank, the personalized PageRank (PPR) (Page et al. 1999) modification was proposed, which avoids this property. We adapt this principle to MPNNs by combining the result of message-passing iterations with the initial node features. We guarantee convergence of node representations to a steady state by using the geometric series of the exponential function. This allows the use of infinitely many message-passing iterations, while ensuring convergence and that the initial node representations affect the final node representations. We introduce an efficient fixed-point iteration scheme with an absolute stopping criterion for computing the final node representations. This chapter is based on Roth and Liebig (2022b).

5.2 Related Work

We now review relevant work on dealing with CD and the connection between MPNNs and PageRank. In Section 5.2.1, we introduce the PageRank algorithm and personalized PageRank (PPR) as a modification, which forms the basis of our approach. We then provide an overview of related graph neural network methods.

5.2.1 PageRank and Personalized PageRank

PageRank (Page et al. 1999) was initially proposed as an algorithm to quantify the importance of web pages. The key idea of PageRank is to assign higher scores to web pages linked to by many other relevant web pages, and a lower score otherwise. In PageRank, the set of n web pages is modeled as a directed graph, where each node represents a web page and each edge corresponds to a link from one web page to another. PageRank utilizes the column-stochastic normalized adjacency matrix $\boldsymbol{A}_{\text{pr}} = \boldsymbol{A}\boldsymbol{D}^{-1} \in \mathbb{R}^{n \times n}$, where $\boldsymbol{A} \in \{0, 1\}$ is the adjacency matrix indicating links, and $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ is the diagonal outdegree matrix. For a simplified version of PageRank, the relevance scores are given by the fixed point of

$$\boldsymbol{r} = \boldsymbol{A}_{\text{pr}}\boldsymbol{r}\,, \tag{5.1}$$

where $\boldsymbol{r} \in \mathbb{R}^n$ is the vector containing the relevance scores. This solution exists as $\boldsymbol{A}_{\text{pr}}$ has a leading eigenvalue $\lambda_1 = 1$ with strictly greater magnitude than all other eigenvalues, and $\boldsymbol{r}$ is a corresponding eigenvector of $\boldsymbol{A}_{\text{pr}}$ (Page et al. 1999). Although $\boldsymbol{r}$ can be computed in closed form, it is commonly obtained efficiently via power iteration using the iterative update step

$$\boldsymbol{r}^{(k)} = \boldsymbol{A}_{\mathrm{pr}}\boldsymbol{r}^{(k-1)}\,, \tag{5.2}$$

initialized with some $\boldsymbol{r}^{(0)} \in \mathbb{R}^n$. As $k \to \infty$, $\boldsymbol{r}^{(k)}$ converges to a scaled version of $\boldsymbol{r}$, i.e., $\lim_{k\to\infty} \boldsymbol{r}^{(k)} = c \cdot \boldsymbol{r}$ for some scalar $c \in \mathbb{R}$ depending on $\boldsymbol{r}^{(0)}$. As a well-established algorithm, PageRank has a rich theoretical foundation, many adaptations, and intuitive interpretations. A common view is the probabilistic interpretation, also called the random surfer model (Page et al. 1999). $\boldsymbol{A}_{\mathrm{pr}}$ represents a stochastic transition matrix on the graph with uniform transition probabilities over outgoing edges from each node. In this interpretation, a surfer randomly clicks links on each webpage. When repeatedly clicking on random links, the PageRank scores correspond to the probability of being at a given web page. With an increased number of clicks or iterations, these probabilities become increasingly independent of the initial web page. The matrix $\boldsymbol{A}_{\mathrm{pr}}$ can also be viewed as a random walk matrix, where k denotes the length of the random walk. For almost every initial distribution $\boldsymbol{r}^{(0)}$, PageRank results in the same stationary distribution $\boldsymbol{r}$. Thus, this version of PageRank does not allow a prior distribution $\boldsymbol{r}^{(0)}$ to influence the resulting scores. To address this limitation, Page et al. (1999) also introduced the PPR algorithm, which introduces a restart probability $\alpha \in (0, 1)$. The relevance scores are equal to the fixed point of the equation

$$\boldsymbol{s} = (1-\alpha) \cdot \boldsymbol{A}_{\mathrm{pr}}\boldsymbol{s} + \alpha \cdot \boldsymbol{r}^{(0)}\,, \tag{5.3}$$

where $\boldsymbol{s} \in \mathbb{R}^n$ is the vector of resulting scores. The parameter α is interpreted as the probability of restarting the random walk, often referred to as a teleportation probability. As a result, $\boldsymbol{s}$ is a combination of random walks of different lengths, each starting with the initial distribution $\boldsymbol{r}^{(0)}$. This becomes more intuitive when reformulating Equation 5.3 as a truncated geometric series applied to vector $\boldsymbol{r}^{(0)}$, i.e.,

$$\boldsymbol{s}^{(k)} = \sum_{i=0}^{k} \alpha \cdot (1-\alpha)^i \cdot \boldsymbol{A}_{\mathrm{pr}}^i \boldsymbol{r}^{(0)}\,. \tag{5.4}$$

The i-th term corresponds to a random walk of length i. As with PageRank, PPR converges when taking the limit of infinitely many iterations, i.e., $\lim_{k\to\infty} \boldsymbol{s}^{(k)} = \boldsymbol{s}$. Since $0 < 1-\alpha < 1$, the contribution of longer random walks diminishes, ensuring convergence.

5.2.2 Personalized PageRank for Graph Neural Networks

We identified two primary approaches at the intersection of PPR and message passing in graph neural networks (GNNs). We will now provide details about these proposed methods.

APPNP.

Klicpera et al. (2019) initially identified the connection between PageRank and graph neural networks (GNNs), noting that the property of losing locality in PageRank occurs similarly in GNNs. Due to this close similarity, they proposed replacing message-passing iterations with the PPR algorithm (Equation 5.3). Given initial node features $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, they define the teleportation state as a node-wise transformed state $\boldsymbol{H}^{(0)} = g(\boldsymbol{X})$, where g is instantiated as a row-wise multi-layer perceptron (MLP) (Amari 1967). Using $\boldsymbol{H}^{(0)}$ as the initial distribution, they apply PPR using iterations of the form

$$\boldsymbol{H}^{(k)} = (1-\alpha) \cdot \boldsymbol{A}_{\text{sym}} \boldsymbol{H}^{(k-1)} + \alpha \cdot \boldsymbol{H}^{(0)}, \tag{5.5}$$

where the symmetrically normalized adjacency matrix $\boldsymbol{A}_{\text{sym}}$ is chosen instead of $\boldsymbol{A}_{\text{pr}}$ in PPR to better align with the graph convolutional network (GCN) (Kipf et al. 2017). Similar to PPR, $\boldsymbol{H}^{(k)}$ converges to a fixed point as $k \to \infty$. They approximate this limit by applying a fixed number of iterations, which they refer to as approximate personalized propagation of neural predictions (APPNP). As with PPR, the parameter $\alpha \in (0,1)$ serves as the restart probability. Notably, APPNP does not contain any learnable parameters in the propagation steps. Consequently, given a fixed $\boldsymbol{H}^{(0)}$, the resulting state is unaffected by the optimization process, limiting the effect of learnable parameters.

Implicit Graph Neural Networks.

The implicit framework (Ghaoui et al. 2021) obtains representations via fixed-point equations. While proposed independently of the connection to PPR, the implicit graph neural network (IGNN) (Gu et al. 2020) defines such a fixed-point equation for graph-structured data. Specifically, they propose the non-linear fixed-point equation

$$\boldsymbol{H} = \phi\left(\boldsymbol{A}_{\text{sym}} \boldsymbol{H} \boldsymbol{W} + \boldsymbol{H}^{(0)}\right) \tag{5.6}$$

where ϕ is a component-wise non-expansive function, $\boldsymbol{A}_{\text{sym}}$ is the symmetrically normalized adjacency matrix, $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ is a feature transformation matrix, and $\boldsymbol{H}^{(0)} = g(\boldsymbol{X})$ is a node-wise transformation of the initial node features $\boldsymbol{X}$. For better intuition, when ϕ is the identity function, Equation (5.6) can be expressed iteratively as

$$\boldsymbol{H}^{(k)} = \sum_{i=0}^{k} \boldsymbol{A}_{\text{sym}}^{i} \boldsymbol{H}^{(0)} \boldsymbol{W}^{i} \tag{5.7}$$

which reveals a geometric series similar to Equation 5.4. As $k \to \infty$, $\boldsymbol{H}^{(k)}$ converges to the fixed-point $\boldsymbol{H} = \lim_{k\to\infty} \boldsymbol{H}^{(k)}$ under conditions we describe in the following.

Gu et al. (2020) prove that a fixed-point solution exists if the largest eigenvalue λ_1 of the matrix $\boldsymbol{W}^{\top} \otimes \boldsymbol{A}_{\text{sym}} \in \mathbb{R}^{nd \times nd}$ defined by the Kronecker product $\otimes$ has magnitude less than one. Since $\boldsymbol{A}_{\text{sym}}$ is a fixed matrix with a unique eigenvalue of maximum magnitude by the Perron-Frobenius Theorem (Berman et al. 1994), Gu et al. (2020) propose to constrain $\boldsymbol{W}$ such that the maximum eigenvalue in magnitude λ_1 satisfies $|\lambda_1^{(\boldsymbol{W})}| < 1$. Because optimization of $\boldsymbol{W}$ with gradient descent does not guarantee $|\lambda_1| < 1$, they use projected gradient descent (Goldstein 1964). After each gradient descent step, the updated parameter matrix $\boldsymbol{W}$ is projected onto the closest matrix in the allowed set of feature transformations, i.e., the set of matrices $\boldsymbol{W}$ for which $|\lambda_1| < 1$. Since this set forms an ℓ_1-ball, efficient projection algorithms exist (Duchi et al. 2008). They propose to obtain $\boldsymbol{H}$ by fixed-point iteration with a stopping criterion.

However, since the space of parameter matrices is restricted to a subspace of $\mathbb{R}^{d \times d}$, the fixed-point solutions are similarly constrained. Projected gradient descent further limits the effectiveness of each optimization step because it may alter the direction away from the steepest descent and reduce the step size. Additionally, the projection increases the complexity of the method and complicates the implementation. It is also challenging to combine this method with other approaches, as choosing different aggregation matrices $\tilde{\boldsymbol{A}}$ instead of $\boldsymbol{A}_{\text{sym}}$ changes the eigenvalue constraints imposed on $\boldsymbol{W}$.

5.2.3 Approaches Related to Component Dominance

Several other modifications to MPNNs have been proposed to prevent information loss in models with many message-passing iterations. Although the connection to PPR and CD is less direct, these approaches target similar limitations of MPNNs. Many methods use residual connections to maintain information from previous states (Bresson et al. 2017; Klicpera et al. 2019; G. Li, Xiong, et al. 2020; M. Chen et al. 2020; G. Li, Müller, et al. 2021; Scholkemper et al. 2025). Other approaches combine node representations from all iterations into joint representations (Xu, C. Li, et al. 2018; Fey 2019). Various gating methods enable models to selectively discard information from neighboring representations (Y. Li et al. 2016; Bresson et al. 2017; Rusch, B. P. Chamberlain, et al. 2023; Dwivedi, Joshi, et al. 2023; Finkelshtein et al. 2024). Other approaches found success in rescaling the feature transformation matrix $\boldsymbol{W}$ (Zhao et al. 2020; Oono et al. 2020). Jin et al. (2022) propose regularizing correlations between feature channels. Following our definition of CD in Definition 3.2, these methods can similarly be interpreted as targeting CD specifically, rather than over-smoothing more broadly.

5.3 Transforming PageRank into an Infinite-Depth Graph Neural Network

We first detail the connection between PageRank and the component dominance (CD) property exhibited by many MPNNs. We then show how PPR can be adapted for MPNNs to ensure locality of node representations and prevent CD. We also provide details on convergence guarantees, how to efficiently obtain node representations, and how to perform optimization efficiently.

5.3.1 MPNNs and PageRank

Many established message-passing operations can be formulated as

$$\boldsymbol{X}^{(k)} = \tilde{\boldsymbol{A}}\boldsymbol{X}^{(k-1)}\boldsymbol{W} \tag{5.8}$$

where $\boldsymbol{X}^{(k-1)} \in \mathbb{R}^{n\times d}$ describes the current state of each node, $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n\times n}$ represents the structure of the graph, possibly with arbitrary edge weights for every connected pair of nodes. The matrix $\boldsymbol{W} \in \mathbb{R}^{d\times d}$ is a linear feature transformation that we assume to be shared across all iterations. Such message-passing operations can be equivalently expressed as the matrix-vector product

$$\operatorname{vec}\left(\tilde{\boldsymbol{A}}\boldsymbol{X}^{(k-1)}\boldsymbol{W}\right) = \boldsymbol{T}\boldsymbol{x}^{(k-1)} \tag{5.9}$$

where $\boldsymbol{x}^{(k-1)} = \operatorname{vec}\left(\boldsymbol{X}^{(k-1)}\right)$ is the vectorized form of the node representations and $\boldsymbol{T} = \boldsymbol{W}^\top \otimes \tilde{\boldsymbol{A}} \in \mathbb{R}^{nd\times nd}$ combines the graph structure and the feature transformation as a single matrix using general properties of the Kronecker product $\otimes$.

In Theorem 3.7, we have proven that all such message-passing steps $\boldsymbol{T}$ exhibit component dominance, as defined in Definition 3.2. CD refers to the phenomenon that a single component of $\boldsymbol{X}^{(0)}$ gets increasingly amplified with more iterations. Specifically, the components of $\boldsymbol{X}^{(0)}$ corresponding to the eigenvalues with maximum magnitude of $\boldsymbol{T}$ get dominantly amplified. We have also provided the connection to power iteration in Section 3.5.4. As Equation (5.9) can be equivalently written as $\boldsymbol{x}^{(k)} = \boldsymbol{T}^k\boldsymbol{x}^{(0)}$ using the matrix power $\boldsymbol{T}^k$, power iteration also describes the property that the state $\boldsymbol{x}^{(k)}$ gets increasingly dominated by the eigenvector of $\boldsymbol{T}$ corresponding to the largest eigenvalue in magnitude.

These phenomena are also closely related to PageRank as given in Equation (5.2). Equation (5.9) can be seen as applying a transition function $\boldsymbol{T}$ to the scores $\boldsymbol{x}^{(k-1)}$, though the entries of $\boldsymbol{T}$ may be negative and may not sum to one for any column. As the eigenvalue with maximal magnitude of $\boldsymbol{T}$ may also be negative and have magnitude greater than one, convergence of $\boldsymbol{X}^{(k)}$ for $k \to \infty$ is typically not given. As such, applying PPR with transition matrix $\boldsymbol{T}$ would also not ensure convergence. The various interpretations of PageRank still help our understanding of the phenomenon. Repeatedly applying a transition function $\boldsymbol{T}$ results in a state given by a random walk of corresponding length, although the transition scores are not probabilistic. These random walks increasingly lose locality of their starting point, and $\boldsymbol{X}^{(k)}$ becomes increasingly independent of the initial representations $\boldsymbol{X}^{(0)}$. Combining this with our theoretical study in Chapter 3 and connection to power iteration in Section 3.5.4, the node representations $\boldsymbol{X}^{(k)}$ increasingly align with the dominant eigenvector of $\boldsymbol{T}$. This leads to a loss of information, as the other components have a diminishing effect on $\boldsymbol{X}^{(k)}$.

5.3.2 Personalized PageRank Graph Neural Network

Given the close connection between PageRank (Page et al. 1999) and applying message-passing iterations, we want to investigate further how we can utilize the benefits of PPR to similarly allow for infinitely many message-passing iterations while retaining local information for every node state. Based on PPR, we want to add a chance to teleport to the initial state, which is equivalent to restarting the message-passing process from the initial state. When ensuring convergence of the message-passing process with depth, we ensure that the initial state always contributes to the final state, which thus ensures locality in the representation of each node. As the personalization matrix, we use the transformed initial state $\boldsymbol{H}^{(0)} = f_\theta(\boldsymbol{X})$, as proposed by Klicpera et al. (2019).

Following Equation (5.9), here we consider the message-passing step $\boldsymbol{T} = \boldsymbol{W}^\top \otimes \tilde{\boldsymbol{A}}$ instead of $\boldsymbol{A}_{\text{pr}}$ as used for PPR as given in Equation (5.4). Equivalently to the PPR reformulation of PageRank, for message-passing operations of the form in Equation (5.8), this would lead to

$$\boldsymbol{H}^{(k)} = (1-\alpha) \cdot \tilde{\boldsymbol{A}}\boldsymbol{H}^{(k-1)}\boldsymbol{W} + \alpha \cdot \boldsymbol{H}^{(0)}\,. \tag{5.10}$$

However, for any fixed α, convergence could not be guaranteed as the largest eigenvalue of $(1-\alpha)(\boldsymbol{W}^\top \otimes \tilde{\boldsymbol{A}})$ may be larger than one. This can cause the representations from longer message-passing iterations to dominate the final representation. The constant initial state $\boldsymbol{H}^{(0)}$ would then have an increasingly small effect on that resulting representation, and we would still have a loss of information. IGNN resolves this by constraining the eigenvalues of $\tilde{\boldsymbol{A}}$ and $\boldsymbol{W}$. We want to allow any unconstrained $\boldsymbol{W}$ and $\tilde{\boldsymbol{A}}$, representing any aggregation function with arbitrary edge weights. While the geometric series given in Equation 5.4 only converges when $\boldsymbol{T} = \boldsymbol{W}^\top \otimes \tilde{\boldsymbol{A}}$ has maximum eigenvalue less than one for any $\alpha \in (0,1)$, there may not exist a fixed $\alpha \in (0,1)$ that guarantees the spectral radius of $(1-\alpha)\boldsymbol{T}$ to be less than one. However, we propose to ensure convergence of $\boldsymbol{T}^k$ using the exponential function formula

$$\operatorname{vec}(\boldsymbol{H}) = \lim_{l\to\infty} \sum_{k=0}^{l} \frac{\boldsymbol{T}^k}{k!} \operatorname{vec}\left(\boldsymbol{H}^{(0)}\right) \tag{5.11}$$

applied to the transformed initial state $\boldsymbol{H}^{(0)} = f_\theta(\boldsymbol{X})$. Note that this convergence holds for any $\boldsymbol{T}$, including any graph structure or aggregation function, any learnable parameters, and even any number of computational graphs.

However, restating Equation (5.10) based on this geometric series cannot be done, as any fixed $\alpha \in \mathbb{R}$ can only lead to exponential decay but not factorial decay. Instead, to achieve the factorial decay $k!$ in Equation 5.11, the chance of teleporting back to the initial state needs to increase with k. Equivalently, the chance of extending the random walk needs to be reduced. We observe that $\frac{1}{k!} = \frac{1}{k\cdot(k-1)\ldots 1}$ can be decomposed into such a product of increasing values. An iteration-dependent coefficient $\alpha^{(k)}$ can achieve this.

Since the longest random walk starts from the first iteration $k = 0$, the corresponding coefficient $\alpha^{(0)}$ should be the smallest to ensure proper weighting. For a fixed number of total iterations, this can be directly done. Let l be the total number of iterations. We define the representations iteratively as

$$\boldsymbol{H}^{(k,l)} = \phi\left(\alpha^{(k,l)}\tilde{\boldsymbol{A}}\boldsymbol{H}^{(k-1,l)}\boldsymbol{W} + \boldsymbol{H}^{(0)}\right) \tag{5.12}$$

where $\alpha^{(k,l)} = \frac{1}{(1+l-k)\epsilon}$ for $k, l \in \mathbb{N}$ aligns with our ideas. We added a tunable parameter $\epsilon \in \mathbb{R}$ to control the decay and speed of convergence. We also added a component-wise activation function ϕ. We refer to this method as the personalized PageRank graph neural network (PPRGNN). By setting ϕ as the identity function and $\epsilon = 1$, we recover Equation 5.11 exactly with $\boldsymbol{H} = \lim_{l\to\infty} \boldsymbol{H}^{(l,l)}$. We formally prove that the sequence $\left(\boldsymbol{H}^{(l,l)}\right)_{l\in\mathbb{N}}$ converges to a unique solution when $l \to \infty$ in the following theorem:

Theorem 5.1. The sequence $\left(\boldsymbol{H}^{(l,l)}\right)_{l\in\mathbb{N}}$ as defined by Equation (5.12) converges to a unique solution as $l \to \infty$ for any $\boldsymbol{H}^{(0,0)} \in \mathbb{R}^{n\times d}$, $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n\times n}$, $\boldsymbol{W} \in \mathbb{R}^{d\times d}$, $\boldsymbol{H}^{(0)} \in \mathbb{R}^{n\times d}$ and $\alpha^{(k,l)} = \frac{1}{(1+l-k)\epsilon}$ with any $\epsilon > 0$, and any Lipschitz continuous activation function ϕ with Lipschitz constant $c < \infty$.

Proof. We first show that $\boldsymbol{H}^{(l,l)}$ converges as $l \to \infty$ by proving that

$$\lim_{l\to\infty} \left\|\boldsymbol{H}^{(l,l)} - \boldsymbol{H}^{(l-1,l-1)}\right\|_F = 0\,. \tag{5.13}$$

We first substitute $\boldsymbol{H}^{(l,l)}$ and $\boldsymbol{H}^{(l-1,l-1)}$ with their definitions, resulting in

$$\begin{aligned}&\left\|\boldsymbol{H}^{(l,l)} - \boldsymbol{H}^{(l-1,l-1)}\right\|_F \\ &\quad= \left\|\phi\left(\alpha^{(l,l)}\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1,l)}\boldsymbol{W} + \boldsymbol{H}^{(0)}\right) - \phi\left(\alpha^{(l-1,l-1)}\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-2,l-1)}\boldsymbol{W} + \boldsymbol{H}^{(0)}\right)\right\|_F.\end{aligned} \tag{5.14}$$

As ϕ is Lipschitz continuous for some c by assumption, we obtain

$$\left\|\boldsymbol{H}^{(l,l)} - \boldsymbol{H}^{(l-1,l-1)}\right\|_F \le c\cdot\left\|\alpha^{(l,l)}\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1,l)}\boldsymbol{W} - \alpha^{(l-1,l-1)}\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-2,l-1)}\boldsymbol{W}\right\|_F. \tag{5.15}$$

As $\tilde{\boldsymbol{A}}$ and $\boldsymbol{W}$ are applied to both terms, these can be factored out jointly using the Kronecker product $\boldsymbol{T} = \boldsymbol{W}^\top \otimes \tilde{\boldsymbol{A}}$ and the vectorized form, resulting in

$$\left\|\boldsymbol{H}^{(l,l)} - \boldsymbol{H}^{(l-1,l-1)}\right\|_F \le c\cdot\left\|\boldsymbol{T}\left(\alpha^{(l,l)}\operatorname{vec}(\boldsymbol{H}^{(l-1,l)}) - \alpha^{(l-1,l-1)}\operatorname{vec}(\boldsymbol{H}^{(l-2,l-1)})\right)\right\|_F. \tag{5.16}$$

As we have $\alpha^{(l,l)} = \alpha^{(l-1,l-1)}$, we further have

$$\left\|\boldsymbol{H}^{(l,l)} - \boldsymbol{H}^{(l-1,l-1)}\right\|_F \le c\cdot\alpha^{(l,l)}\cdot\|\boldsymbol{T}\|_F\cdot\left\|\boldsymbol{H}^{(l-1,l)} - \boldsymbol{H}^{(l-2,l-1)}\right\|_F \tag{5.17}$$

More generally, we also have $\alpha^{(l-k,l)} = \alpha^{(l-k-1,l-1)} = \frac{1}{(1+k)\epsilon}$. By iterative application, we obtain

$$\left\|\boldsymbol{H}^{(l,l)} - \boldsymbol{H}^{(l-1,l-1)}\right\|_F \le c^{l-1}\cdot\prod_{k=0}^{l-2}\alpha^{(l-k,l)}\cdot\|\boldsymbol{T}\|_F^{l-1}\cdot\left\|\boldsymbol{H}^{(1,l)} - \boldsymbol{H}^{(0,l-1)}\right\|_F. \tag{5.18}$$

As $\lim_{l\to\infty}\prod_{k=0}^{l-2}\alpha^{(l-k,l)}\|\boldsymbol{T}\|_F^{l-1} = \lim_{l\to\infty}\frac{1}{(l-1)!\epsilon^l}\|\boldsymbol{T}\|_F^{l-1} = 0$ and for any $\boldsymbol{H}^{(0,l-1)}$ and $\boldsymbol{H}^{(0,l)}$ there exists a $c_2 \in \mathbb{R}$ for which $\left\|\boldsymbol{H}^{(1,l)} - \boldsymbol{H}^{(0,l-1)}\right\|_F \le c_2$. This proves the convergence of the sequence.

To prove the uniqueness of the solution, it suffices to note that the above proof of convergence holds independently of the initial states $\boldsymbol{H}^{(0,l-1)}$ and $\boldsymbol{H}^{(0,l)}$. $\square$

As we have established convergence of the PPRGNN when taking infinitely many iterations, there exists a limit representation $\boldsymbol{H} = \lim_{l\to\infty} \boldsymbol{H}^{(l,l)}$. We can obtain $\boldsymbol{H}$ using fixed-point iteration with an absolute stopping criterion as the representation $\boldsymbol{H}^{(l,l)}$ with index

$$l = \min_{k\in\mathbb{N}} \left\| \boldsymbol{H}^{(k,k)} - \boldsymbol{H}^{(k-1,k-1)} \right\| < \gamma \tag{5.19}$$

for some convergence threshold $\gamma > 0$. However, we cannot utilize any intermediate results of computing $\boldsymbol{H}^{(k-1,k-1)}$ to compute $\boldsymbol{H}^{(k,k)}$, as it requires a full recomputation, each $\boldsymbol{H}^{(k,k)}$ having complexity linear in k. Instead, we approximate l by determining at which length a term $\frac{\boldsymbol{T}^k}{k!}\operatorname{vec}(\boldsymbol{X}^{(0)})$ has a negligible impact on the result. We do this by ignoring the restart term $\boldsymbol{H}^{(0)}$ and considering the recursive equation

$$\boldsymbol{G}^{(k)} = \phi\left(\alpha^{(1,k)}\tilde{\boldsymbol{A}}\boldsymbol{G}^{(k-1)}\boldsymbol{W}\right) \tag{5.20}$$

with $\boldsymbol{G}^{(0)} = \boldsymbol{H}^{(0)}$. The benefit of this is that it can be computed independently of l and thus requires a single forward pass. We formally prove the convergence of $\boldsymbol{G}^{(k)}$ as $k \to \infty$ in the following proposition:

Proposition 5.1. Let $\boldsymbol{G}^{(k)} = \phi\left(\alpha^{(1,k)}\tilde{\boldsymbol{A}}\boldsymbol{G}^{(k-1)}\boldsymbol{W}\right)$ with $\alpha^{(1,k)} = \frac{1}{k\epsilon}$ for any $\epsilon > 0$, any $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n\times n}$, $\boldsymbol{W} \in \mathbb{R}^{d\times d}$, $\boldsymbol{G}^{(0)} \in \mathbb{R}^{n\times d}$ and component-wise non-expansive activation function ϕ. Then,

$$\lim_{l\to\infty} \left\| \boldsymbol{G}^{(l)} \right\|_F = 0 \tag{5.21}$$

Proof. Based on the definition $\boldsymbol{G}^{(l)} = \phi\left(\alpha^{(1,l)}\tilde{\boldsymbol{A}}\boldsymbol{G}^{(l-1)}\boldsymbol{W}\right)$, we first use the fact that $\|\phi(\boldsymbol{X})\|_F \leq \|\boldsymbol{X}\|_F$ for any component-wise non-expansive function ϕ, resulting in

$$\left\| \boldsymbol{G}^{(l)} \right\|_F \leq \left\| \alpha^{(1,l)}\tilde{\boldsymbol{A}}\boldsymbol{G}^{(l-1)}\boldsymbol{W} \right\|_F. \tag{5.22}$$

Using the sub-multiplicativity of the Frobenius norm, we obtain

$$\left\| \boldsymbol{G}^{(l)} \right\|_F \leq \alpha^{(1,l)} \cdot \|\boldsymbol{W}\|_F \cdot \left\| \tilde{\boldsymbol{A}} \right\|_F \cdot \left\| \boldsymbol{G}^{(l-1)} \right\|_F. \tag{5.23}$$

By iterative application, we have

$$\left\| \boldsymbol{G}^{(l)} \right\|_F \leq \prod_{k=1}^{l} \alpha^{(1,k)} \cdot \|\boldsymbol{W}\|_F^l \cdot \left\| \tilde{\boldsymbol{A}} \right\|_F^l \cdot \left\| \boldsymbol{G}^{(0)} \right\|_F. \tag{5.24}$$

By definition of $\alpha^{(1,k)}$, their product is given as $\prod_{k=1}^{l} \alpha^{(1,k)} = \frac{1}{l!\,\epsilon^l}$. In the limit case, this results in

$$\lim_{l\to\infty} \frac{\|\boldsymbol{W}\|_F^l \cdot \left\| \tilde{\boldsymbol{A}} \right\|_F^l}{l!\,\epsilon^l} \cdot \left\| \boldsymbol{G}^{(0)} \right\|_F = 0 \tag{5.25}$$

for any $\boldsymbol{G}^{(0)}, \boldsymbol{W}, \tilde{\boldsymbol{A}}$ and $\epsilon > 0$ due to the factorial growing faster than the exponential. □

Based on the adapted random walk $\boldsymbol{G}^{(l)}$ of length l having negligible impact on the resulting representations, we approximate the index l used within the fixed-point iteration point as

$$l \approx \min_{k \in \mathbb{N}} \left\| \boldsymbol{G}^{(k)} \right\| < \gamma \,. \tag{5.26}$$

In combination, we approximate $\boldsymbol{H}$ by iterating Equation (5.20) to approximate l based on Equation (5.26), we then iterate Equation (5.12) to obtain $\boldsymbol{H}^{(l,l)}$.

5.3.3 Efficient Optimization

While our method to approximate $\boldsymbol{H}$ requires iterating $\boldsymbol{G}^{(l)}$ until convergence and computing $\boldsymbol{H}^{(l,l)}$, the optimization process can be performed efficiently. Since the state $\boldsymbol{G}^{(l)}$ is not used for predictions, we do not need to track the gradients of this computation, making the training process significantly more efficient. However, as the number of iterations l is not known or constrained, the required memory for computing gradients for $\boldsymbol{H}$ is unconstrained and may be time-consuming. We will now propose a method to obtain gradients efficiently with bounded memory consumption. We are interested in the partial derivative of the resulting error $\ell(\boldsymbol{H}) \in \mathbb{R}$ obtained by some loss function ℓ with respect to the parameter matrix $\boldsymbol{W}$ and the initial state $\boldsymbol{H}^{(0)}$. Based on the approximation $\boldsymbol{H}^{(l,l)} \approx \boldsymbol{H}$, we obtain the partial derivative with respect to the input $\boldsymbol{H}^{(0)}$ as

$$\frac{\partial \ell\left(\boldsymbol{H}^{(l,l)}\right)}{\partial \boldsymbol{H}^{(0)}} = \sum_{k=0}^{l} \frac{\partial \ell(\boldsymbol{H}^{(l,l)})}{\partial \boldsymbol{Z}^{(l-k,l)}} \tag{5.27}$$

with $\boldsymbol{Z}^{(0,l)} = \boldsymbol{H}^{(0)}$ and $\boldsymbol{Z}^{(k,l)} = \alpha^{(k,l)} \tilde{\boldsymbol{A}} \boldsymbol{H}^{(k-1,l)} \boldsymbol{W} + \boldsymbol{H}^{(0)}$ so that $\boldsymbol{H}^{(k,l)} = \phi(\boldsymbol{Z}^{(k,l)})$. We obtained the partial derivative with respect to the parameters $\boldsymbol{W}$ as

$$\frac{\partial \ell\left(\boldsymbol{H}^{(l,l)}\right)}{\partial \boldsymbol{W}} = \sum_{k=0}^{l} \left(\alpha^{(l-k,l)} \left(\boldsymbol{H}^{(l-k-1,l)} \right)^{\top} \tilde{\boldsymbol{A}}^{\top} \right) \frac{\partial \ell\left(\boldsymbol{H}^{(l,l)}\right)}{\partial \boldsymbol{Z}^{(l-k,l)}} \,. \tag{5.28}$$

To obtain these, we further need to compute the partial derivative $\frac{\partial \ell(\boldsymbol{H}^{(l,l)})}{\partial \boldsymbol{Z}^{(k,l)}}$, which we obtained as

$$\frac{\partial \ell\left(\boldsymbol{H}^{(l,l)}\right)}{\partial \boldsymbol{Z}^{(k,l)}} = \frac{\partial \ell\left(\boldsymbol{H}^{(l,l)}\right)}{\partial \boldsymbol{H}^{(k,l)}} \odot \phi' \left(\boldsymbol{Z}^{(k,l)} \right) \,, \tag{5.29}$$

where $\phi'(\cdot)$ is the element-wise derivative of the activation function ϕ applied to its input. This in turn requires the partial derivative $\frac{\partial \ell(\boldsymbol{H}^{(l,l)})}{\partial \boldsymbol{H}^{(k-1,l)}}$, which requires the following two cases:

$$\frac{\partial \ell\left(\boldsymbol{H}^{(l,l)}\right)}{\partial \boldsymbol{H}^{(k,l)}} = \begin{cases} \alpha^{(k+1,l)} \tilde{\boldsymbol{A}}^{\top} \left(\frac{\partial \ell(\boldsymbol{H}^{(l,l)})}{\partial \boldsymbol{Z}^{(k+1,l)}} \right) \boldsymbol{W}^{\top}, & \text{if } k < l \\ \frac{\partial \ell(\boldsymbol{H}^{(l,l)})}{\partial \boldsymbol{H}^{(l,l)}}, & \text{if } k = l \,. \end{cases} \tag{5.30}$$

Dataset	# of graphs	avg. # of nodes	# of classes	Task
PPI	22	2373	121	Inductive Node classification
Amazon	1	334 863	58	Transductive Node classification
MUTAG	188	17.9	2	Graph classification
PTC	344	25.5	2	Graph classification
COX2	467	41.2	2	Graph classification
PROTEINS	1113	39.1	2	Graph classification
NCI1	4110	29.8	2	Graph classification

Table 5.1: Key properties of the graphs used for our evaluation.

For $l \to \infty$, both Equation 5.27 and Equation 5.28 converge to a unique solution, as with the forward pass. For efficient computation of these partial derivatives, we use gradient checkpointing (T. Chen et al. 2016) to store intermediate results $\boldsymbol{H}^{(k,l)}$ and $\boldsymbol{Z}^{(k,l)}$ during the forward pass. To additionally constrain the memory consumption during training for the intermediate states for backpropagation, we further constrain the maximal k used within the sums by a tunable number of iterations m instead of summing over all l terms. The number m can be selected based on the available memory and runtime constraints. This is similar to truncated backpropagation through time (TBPTT) (Elman 1990), which was proposed as a memory-efficient optimization procedure for recurrent neural networks. This truncation also reduces the runtime of the backpropagation. As the representations of each node in $\boldsymbol{H}^{(l,l)}$ are unaffected by the representations of other nodes with distance larger than l hops in the graph, the partial derivatives would be unaware of this potential information. To account for this, we compute the partial derivatives for $l' = l + j$ instead of setting l as the maximal index. This allows optimization to account for further distant representations that would otherwise be unnoticeable. Thus, we compute the partial derivative $\frac{\partial \ell(\boldsymbol{H}^{(l',l')})}{\partial \boldsymbol{H}^{(0)}}$ and $\frac{\partial \ell(\boldsymbol{H}^{(l',l')})}{\partial \boldsymbol{W}}$.

In summary, these approximations enable a more efficient optimization process in terms of runtime and memory usage by truncating the number of backpropagation steps and utilizing gradient checkpointing.

5.4 Evaluation

We now empirically validate the effectiveness of the PPRGNN using several commonly used benchmark datasets. We follow the experiments conducted for evaluating the IGNN (Gu et al. 2020). We utilize the implementation provided with the IGNN and solely replace their infinite-depth formulation with our proposed formulation. Equivalently to their approach, we evaluate a model based on the GCN and set the aggregation function $\tilde{\boldsymbol{A}} = \boldsymbol{A}_{\text{sym}}$ to the symmetrically normalized adjacency matrix. PPRGNN and IGNN utilize

Method	Micro-F_1-Score
MLP	46.2
GCN	59.2
SSE	83.6
GAT	97.3
APPNP	44.8
IGNN	97.6
PPRGNN	**98.9**

Table 5.2: Obtained micro-F_1-scores on the testing set. Best result in **bold**, second-best result underlined.

Table 5.3: Runtimes and required number of optimization steps for the IGNN and PPRGNN on the PPI and Amazon tasks. Both methods use the same architecture and the same number of parameters. The # of epochs for PPRGNN is the epoch after which PPRGNN surpasses the IGNN in performance. Total time refers to the runtime required for this performance.

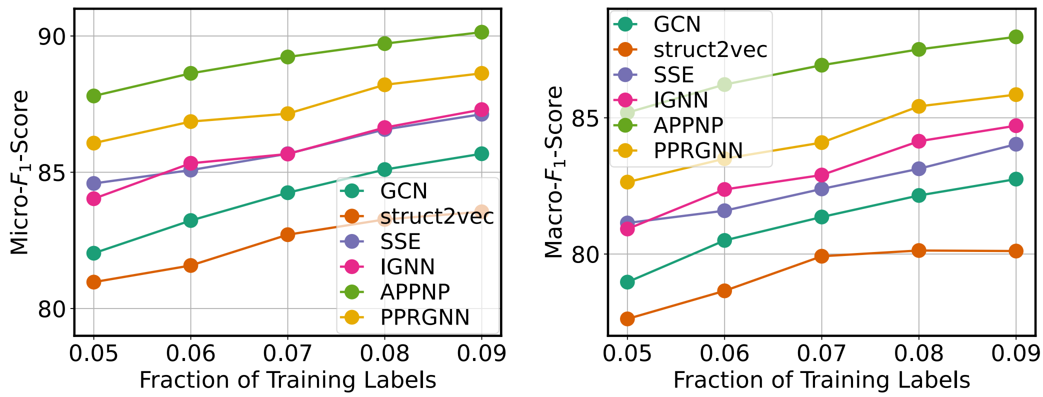
Dataset	Method	# of Epochs	Avg. time per epoch	Total time
Amazon (0.05)	IGNN	872	14s	3h 21m
	PPRGNN	**175**	**11s**	**32m**
PPI	IGNN	58	26s	25m
	PPRGNN	**47**	**18s**	**14m**

the same number of parameters. We additionally evaluate APPNP (Klicpera et al. 2019) by equivalently replacing the IGNN with their proposed formulation given in Equation (5.5). Following the experiments conducted to evaluate the IGNN, we consider the protein-protein interactions (PPI) benchmark dataset (Zitnik et al. 2017), the Amazon benchmark dataset (J. Yang et al. 2015), and five graph classification datasets, namely MUTAG, PTC, COX2, PROTEINS, and NCI1 (Yanardag et al. 2015). For the backward pass, we use a maximum number of $m = 5$ iterations. All experiments use a fixed initial learning rate, which we decay when the training loss plateaus for 24 epochs. All experiments were executed on an Nvidia P100 GPU. Our implementation is publicly available [5] .

PPI

As the first task, we consider the protein-protein interactions (PPI) benchmark dataset (Zitnik et al. 2017). Each of the 22 graphs corresponds to a specific human tissue, with each node representing a protein and each edge indicating an interaction between two proteins. Based on 50 available features for each node, the task is to perform binary classification for each of 121 available classes representing cellular functions. Matching

[5] `https://github.com/roth-andreas/pprgnn`



(a) Micro-F_1-Scores. **(b)** Macro-F_1-Scores.

Figure 5.1: Testing prediction results for the Amazon task using micro-F_1-scores (a) and macro-F_1-scores (b). The fraction indicates the share of labeled nodes available for optimization.

the typical evaluation procedure for this task and the implementation of the IGNN, we utilize 18 graphs for training, two graphs for validation, and two graphs for testing. We evaluate the architecture used by the IGNN, which applies five separate infinite-depth iterations in sequence. We find that removing self-loops is beneficial for this task, and we set $\epsilon = 0.25$. Based on the results from IGNN, we also provide their results for an MLP, the graph convolutional network (Kipf et al. 2017), steady-state embedding (SSE) (Dai et al. 2018), and the graph attention network (Velickovic et al. 2018).

The micro-F_1-scores for the testing graphs are presented in Table 5.2. While the IGNN achieved an F_1-score of 97.6, the PPRGNN was able to improve this result to 98.9. The APPNP underfits the data and achieves an F_1-score of 44.8. This holds even when the initial feature transformation before applying the APPNP uses significantly more parameters than the PPRGNN and IGNN. This improvement supports the need for a more complex model with task-specific parameters. GAT achieves a slightly lower F_1-score compared to the IGNN at 97.3, but significantly better than the GCN. In Table 5.3, we additionally compare runtimes for the IGNN and the PPRGNN. The average runtime per epoch for the PPRGNN is around 18 seconds compared to 26 seconds for the IGNN, an improvement of more than 20%. It also achieves the lowest validation error after 47 epochs, an improvement of around 15% compared to 58 epochs for the IGNN. The optimized PPRGNN utilizes a total of 82 message-passing iterations for the testing graphs over the five sequential runs until convergence.

Amazon

Following the evaluation of the IGNN, we now assess the scalability of the PPRGNN on the Amazon product co-purchasing dataset (J. Yang et al. 2012), which has been preprocessed according to Dai et al. (2018). Each of the 334 863 nodes in the graph represents

Table 5.4: Evaluation of the PPRGNN for five node classification tasks, following (Gu et al. 2020). Best results in **bold**, second-best <u>underlined</u>.

Dataset	MUTAG	PTC	COX2	PROTEINS	NCI1
GK	81.4 ± 1.7	55.7 ± 0.5	-	71.4 ± 0.3	62.5 ± 0.3
RW	79.2 ± 2.1	55.9 ± 0.3	-	59.6 ± 0.1	-
WL	84.1 ± 1.9	58.0 ± 2.5	83.2 ± 0.2	74.7 ± 0.5	$\mathbf{84.5} \pm 0.5$
DGCNN	85.8	58.6	-	75.5	74.4
GCN	85.6 ± 5.8	64.2 ± 4.3	-	76.0 ± 3.2	80.2 ± 2.0
GIN	89.0 ± 6.0	63.7 ± 8.2	-	75.9 ± 3.8	82.7 ± 1.6
IGNN	$\underline{89.3} \pm 6.7$	$\underline{70.1} \pm 5.6$	$\underline{86.9} \pm 4.0$	77.7 ± 3.4	80.5 ± 1.9
APPNP	87.7 ± 8.6	64.5 ± 5.1	82.2 ± 5.5	$\underline{78.7} \pm 4.8$	65.9 ± 2.7
PPRGNN	$\mathbf{90.4} \pm 7.2$	$\mathbf{75.0} \pm 5.7$	$\mathbf{89.1} \pm 3.9$	$\mathbf{80.2} \pm 3.2$	$\underline{83.5} \pm 1.5$

a product, and every edge indicates that the two corresponding products have been purchased together. In this transductive node classification task, the task is to predict one out of 58 product types for each node based on a small number of labeled products. There are no node features provided. The fraction of labeled products varies between 5% and 9%, as proposed by Dai et al. (2018). Following the model structure of the IGNN, a single PPRGNN layer is used, with a linear encoder and linear decoder used before and after. As in the IGNN experiments, we also present results of the SSE (Dai et al. 2018), struc2vec (Ribeiro et al. 2017), GCN (Kipf et al. 2017). We additionally evaluate the APPNP for this task. Micro F_1 and Macro F_1 scores are visualized in Figure 5.1 for label fractions between 5% and 9%. PPRGNN outperforms the IGNN, SSE, GCN, and struc2vec by at least 1% in all cases. In contrast to the more complex PPI dataset, APPNP outperforms PPRGNN and all other methods. This indicates that the learnable parameters are less critical for this task compared to the PPI dataset. In Table 5.3, we also present runtimes for the IGNN and the PPRGNN on this dataset. As with the runtimes for the PPI task, the average time per epoch is reduced from 14 seconds to 11 seconds. Even more critically, the optimization requires fewer steps, as the minimal validation error is achieved after 175 epochs for the PPRGNN, compared to 872 epochs for the IGNN. This may be caused by the inefficiency of the projected gradient descent used within the IGNN. Both factors combined lead to a significantly reduced total training time.

Graph Classification

We now consider the five graph classification tasks MUTAG, PTC, COX2, PROTEINS, and NCI1 (Morris, Kriege, Bause, et al. 2020). These are relatively small standard benchmark datasets for binary classification of graphs. We provide key properties of these datasets in Table 5.1. Following the setup of the IGNN, we conduct a ten-fold cross-validation and report the mean and standard deviation across the validation sets. Based on their experimental setup, the employed architecture applies three infinite-depth iterations in sequence. We set some joint hyperparameters across all five datasets: The convergence threshold is set to $\epsilon = 1$, weight decay is set to $1e - 6$, and gradient clipping is set to 25. Apart from NCI1, self-loops were found to be generally beneficial and added to the model. As in IGNN, we present several baselines, including classical kernel methods. These include the random walk (RW) kernel (Gärtner et al. 2003), the Weisfeiler-Leman (WL) kernel (Shervashidze, Schweitzer, et al. 2011), and the graphlet kernel (GK) (Shervashidze, Vishwanathan, et al. 2009). We also present results for the GCN (Kipf et al. 2017), the graph isomorphism network (GIN) (Xu, Hu, et al. 2019), and the deep graph convolutional neural network (DGCNN) (Zhang et al. 2018).

The average results are presented in Table 5.4. The PPRGNN achieves the best results for the MUTAG, PTC, COX2, and PROTEINS datasets. For the NCI1 dataset, it achieves the second-best result, as the WL kernel achieves a higher accuracy by 1%. After training, the utilized depth of the PPRGNN varies between 22 and 41 iterations before the convergence threshold is satisfied. These experiments confirm that all methods avoiding CD, PPRGNN, IGNN, and APPNP, improve results compared to the GCN significantly for many tasks. Compared to the IGNN and APPNP, the PPRGNN also consistently achieves promising results.

6 Conclusion and Future Work

In this thesis, we studied properties exhibited by message-passing neural networks (MPNNs) theoretically and empirically. We extended existing insights on MPNNs by identifying several properties exhibited by many established methods. These split the previously identified phenomenon of over-smoothing into separate, finer-grained properties. These findings are not only beneficial for an improved understanding of MPNNs but also allow the streamlined design of methods that exhibit desired properties. We also propose several frameworks for MPNNs, each specifically targeted to avoid one of the properties.

With these findings, we advance the field of graph machine learning in several ways. Our improved understanding of MPNNs can lead to more targeted and theoretically grounded future research. Our research seeks to assist in addressing some of the ongoing challenges and open up new opportunities.

In Section 6.1, we summarize our main findings and relate them to the initial research question stated in Section 1.2. In Section 6.2, we outline how our contributions can affect future work on graph machine learning and present essential follow-up questions.

6.1 Conclusion

This thesis presented several contributions related to two research questions outlined in Section 1.2, structured into three main chapters. Following Research Question 1, we studied the effect of message-passing operations on given node representations. We demonstrated that the commonly observed phenomenon of over-smoothing in MPNNs can be decomposed into multiple underlying, finer-grained properties. First, we identified that the norm of the representations can vanish under certain conditions of the feature transformation, a phenomenon that overlaps with over-smoothing. This causes all node representation vectors to converge towards the zero vector, which can lead to over-smoothing under specific definitions and metrics. By further studying the effect of message-passing iterations, we identified that for many MPNNs, each single message-passing operation exhibits the shared component amplification (SCA) property. SCA refers to each message-passing iteration amplifying the same component in the data for every feature channel, independently of the parameter instantiation. This constrains the set of representable node representations when applying a message-passing iteration, and limits the effect of the learnable parameters and the optimization. While closely related to over-smoothing, SCA underlies it and describes a part that occurs for each individual message-passing iteration, rather than only for multiple iterations or in the limit. For the limit case of infinitely many message-passing iterations, we identified that a single component in the data is dominantly amplified. In contrast, all other components are relatively dampened increasingly, a property we refer to as component dominance (CD). MPNNs exhibiting the CD property yield node representations that become increasingly less informative and independent of the initially provided node representations. We additionally provide spectral intuition of SCA and CD by showing that message-passing iterations can be interpreted as filters of the spectral graph convolution in the graph Fourier domain. We also show that MPNNs can be interpreted as a special case of the power iteration algorithm. Similarly to CD, power iteration describes the convergence of a vector to the dominant eigenvector of a matrix when repeatedly applying that matrix.

Following our novel insights into the effect of message-passing on representations, we then considered our other challenge. With Research Question 2, our goal was to design MPNNs with beneficial properties based on our improved understanding of these methods. Specifically, we developed frameworks for MPNNs that do not exhibit either the SCA or the CD property.

To design MPNNs that do not exhibit SCA, we proved the necessity of operating on multiple computational graphs. Equivalently, we require multiple edge relations in a graph, e.g., a multigraph or a multi-relational graph. We proposed two approaches for deriving such graphs. First, we introduced the multi-relational split (MRS) framework, with which we proposed converting the computational graph of any existing MPNN into multiple edge relations. Each edge is assigned to one of multiple edge relation types, for each of which a distinct feature transformation is applied to the corresponding message. We introduced structural dependence and independence of nodes, which allowed us to identify cases in which the SCA property can be avoided. While the construction of a method for splitting the computational graph is highly task-dependent, our identified property holds independently of the specific method employed. As a general method that always ensures the structural independence of nodes, we propose splitting a graph into two directed acyclic relations using a strict partial ordering. We demonstrate that this can be efficiently performed, for example, by utilizing the node degree for ordering and distinguishing between edges from higher-degree nodes to lower-degree nodes and vice versa.

As an alternative derivation of multiple computational graphs, we proposed a novel approximation of message-passing methods based on the spectral graph convolution. While previously the graph convolution has only been defined in the single-input single-output (SISO) case, we obtained the multi-input multi-output graph convolution (MIMO-GC) using the convolution theorem and the graph Fourier transform. We found that the MIMO-GC naturally utilizes multiple computational graphs. Each computational graph amplifies a distinct component in the representation, matching our theoretical findings on the SCA property. We proposed the localized MIMO-GC (LMGC) as a framework for message-passing, which approximates the MIMO-GC by localizing the aggregation step while allowing for multiple computational graphs. The LMGC framework only requires the construction of edge weights, streamlining the design of message-passing methods and the study of their properties. We present two key properties of the LMGC framework. For almost every choice of edge weights, the LMGC is injective on multisets, similar to the graph isomorphism network (GIN) (Xu, Hu, et al. 2019). This property holds even for a single computational graph. For multiple computational graphs, the resulting representations are linearly independent, a property that subsumes injectivity. Our findings also highlighted the close connections between the expressivity of message-passing iterations and avoiding SCA as one key property underlying over-smoothing.

To develop methods that avoid CD, we established a close connection between this phenomenon and the PageRank algorithm. This connection helps to provide a better intuition for CD, as message-passing iterations can be seen as random walk steps in PageRank. It also allows us to adopt modifications that avoid this phenomenon. As personalized PageRank (PPR) was proposed as a modification to PageRank, we proposed adapting PPR for MPNNs and constructing a method that follows its ideas. As PPR ensures the localization of the resulting state by combining multiple random walks while guaranteeing the convergence of each random walk, we proposed to ensure the convergence of message-passing iterations. We refer to the resulting method as the personalized PageRank graph neural network (PPRGNN), whic avoids CD by similarly ensuring convergence of MPNNs to a fixed point. We also proposed an efficient method for obtaining the fixed point, both in terms of runtime and memory consumption.

6.2 Future Work

The differences between most recent methods on established benchmark tasks are relatively minor (Bechler-Speicher et al. 2025). This lack of challenging benchmark tasks has limited the informativeness of our empirical evaluations. These datasets do not allow an effective examination of the importance of avoiding specific properties in message-passing. To enable more effective empirical evaluations, future work is needed to develop meaningful real-world benchmark applications for graph machine learning methods. When performance improvements better reflect the practical relevance of a method, promising research directions can be identified more effectively. Currently, weighing the importance of each property is a challenging task. While various phenomena of MPNNs have been studied both empirically and theoretically, the relevance of addressing any particular issue remains uncertain, as also noted by Arnaiz-Rodríguez et al. (2025). This challenge is intensified by the fact that many of these phenomena lack clear definitions and refer to broad aspects. Disentangling observed phenomena into fine-grained properties is crucial for aligning research efforts, identifying critical research directions, and developing more targeted solutions.

Following our novel insights on message-passing iterations, future work can focus on targeting fine-grained properties that underlie over-smoothing, rather than attempting to avoid the broad phenomenon as a whole. Previous work on over-smoothing has addressed different aspects of the property, without recognizing that these aspects are distinct. Our identification of over-smoothing as composed of multiple overlapping phenomena allows the separation of these methods. The broad scope of over-smoothing not only complicates efforts to establish a unified definition and understanding of the phenomenon, but it also makes it difficult to prioritize which aspects are most critical to address. By splitting over-smoothing into multiple separate properties, it becomes easier to design specific methods targeting particular fine-grained properties. It also makes it easier to study which underlying aspects of over-smoothing are most relevant to a given task. We see considerable potential for future work to further identify finer-grained properties of message-passing methods, enabling the field to pinpoint the most critical properties. Our theoretical insights can be further extended by identifying cases in which avoiding SCA, CD, or other aspects of message-passing is most beneficial. The theoretical insights can also be extended to more methods, including nonlinear activation functions for properties that consider iterated message-passing.

Regarding SCA, further research could explore the design of methods using multiple computational graphs. In addition to our identified general properties of such frameworks, additional relevant aspects can be identified. As tasks may benefit from different methods depending on the available data, task-specific methods with particular properties can be constructed. Although our empirical results demonstrate the benefits of avoiding SCA for various tasks, its relative importance in contrast to other phenomena requires further investigation. Similarly, our insights for avoiding CD can be further extended in future work by analyzing its importance in comparison with other properties of MPNNs. Building on our insights into the connection between MPNNs and PageRank, similar methods can be developed that further mitigate CD.

Overall, the fragmented understanding of various phenomena and their significance has resulted in disjointed research efforts in graph machine learning. By reaching a shared understanding of key phenomena, research efforts could be more streamlined and targeted. Once the community reaches consensus on clearly defined aspects and their relevance in graph machine learning, the field could benefit substantially from more targeted research. This will also require the development of challenging and impactful benchmarks, along with a systematic evaluation of the importance of various properties for performance improvements and research prioritization.

List of Figures

List of Tables

List of Footnotes and Online Resources

Bibliography

Abboud, Ralph, Radoslav Dimitrov, and İsmail İlkan Ceylan (2022). "Shortest Path Networks for Graph Property Prediction". In: *Learning on Graphs Conference, LoG 2022, 9-12 December 2022, Virtual Event*, p. 5.

Alon, Uri and Eran Yahav (2021). "On the Bottleneck of Graph Neural Networks and its Practical Implications". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021.*

Amari, Shun-ichi (1967). "A Theory of Adaptive Pattern Classifiers". In: *IEEE Trans. Electron. Comput.*, pp. 299–307. DOI: 10.1109/PGEC.1967.264666.

Arnaiz-Rodríguez, Adrián and Federico Errica (2025). "Oversmoothing, "Oversquashing", Heterophily, Long-Range, and more: Demystifying Common Beliefs in Graph Machine Learning". In: *CoRR*. DOI: 10.48550/ARXIV.2505.15547.

Babai, László (2016). "Graph isomorphism in quasipolynomial time [extended abstract]". In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pp. 684–697. DOI: 10.1145/2897518.2897542.

Barbero, Federico, Ameya Velingker, Amin Saberi, Michael M. Bronstein, and Francesco Di Giovanni (2024). "Locality-Aware Graph Rewiring in GNNs". In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024.*

Bechler-Speicher, Maya, Ben Finkelshtein, Fabrizio Frasca, Luis Müller, Jan Tönshoff, Antoine Siraudin, Viktor Zaverkin, Michael M. Bronstein, Mathias Niepert, Bryan Perozzi, Mikhail Galkin, and Christopher Morris (2025). "Position: Graph Learning Will Lose Relevance Due To Poor Benchmarks". In: *CoRR*. DOI: 10.48550/ARXIV.2502.14546.

Bengio, Yoshua, Patrice Y. Simard, and Paolo Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Trans. Neural Networks*, pp. 157–166. DOI: 10.1109/72.279181.

Berman, Abraham and Robert J. Plemmons (1994). *Nonnegative Matrices in the Mathematical Sciences*. SIAM. DOI: 10.1137/1.9781611971262.

Bo, Deyu, Xiao Wang, Chuan Shi, and Huawei Shen (2021). "Beyond Low-frequency Information in Graph Convolutional Networks". In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in*

Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, pp. 3950–3957. DOI: 10.1609/AAAI.V35I5.16514.

Bodnar, Cristian, Francesco Di Giovanni, Benjamin Paul Chamberlain, Pietro Lió, and Michael M. Bronstein (2022). "Neural Sheaf Diffusion: A Topological Perspective on Heterophily and Oversmoothing in GNNs". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022.*

Borgwardt, Karsten M. and Hans-Peter Kriegel (2005). "Shortest-Path Kernels on Graphs". In: *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA*, pp. 74–81. DOI: 10.1109/ICDM.2005.132.

Bouritsas, Giorgos, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein (2023). "Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting". In: *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 657–668. DOI: 10.1109/TPAMI.2022.3154319.

Bresson, Xavier and Thomas Laurent (2017). "Residual Gated Graph ConvNets". In: *CoRR*.

Brody, Shaked, Uri Alon, and Eran Yahav (2022). "How Attentive are Graph Attention Networks?" In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022.*

Bruna, Joan, Wojciech Zaremba, Arthur Szlam, and Yann LeCun (2014). "Spectral Networks and Locally Connected Networks on Graphs". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings.*

Butler, Landon, Alejandro Parada-Mayorga, and Alejandro Ribeiro (2023). "Convolutional Learning on Multigraphs". In: *IEEE Trans. Signal Process.*, pp. 933–946. DOI: 10.1109/TSP.2023.3259144.

Cai, Chen and Yusu Wang (2018). "A simple yet effective baseline for non-attribute graph classification". In: *CoRR*.

Cai, Chen and Yusu Wang (2020). "A Note on Over-Smoothing for Graph Neural Networks". In: *CoRR*.

Cao, Jian, Marc G. Genton, David E. Keyes, and George M. Turkiyyah (2021). "Sum of Kronecker products representation and its Cholesky factorization for spatial covariance matrices from large grids". In: *Comput. Stat. Data Anal.*, p. 107165. DOI: 10.1016/J.CSDA.2020.107165.

Chen, Deli, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun (2020). "Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View". In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020,* *New York, NY, USA, February 7-12, 2020*, pp. 3438–3445. DOI: 10.1609/AAAI.V34I04.5747.

Chen, Ming, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li (2020). "Simple and Deep Graph Convolutional Networks". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, pp. 1725–1735.

Chen, Tianqi, Bing Xu, Chiyuan Zhang, and Carlos Guestrin (2016). "Training Deep Nets with Sublinear Memory Cost". In: *CoRR*.

Chien, Eli, Jianhao Peng, Pan Li, and Olgica Milenkovic (2021). "Adaptive Universal Generalized PageRank Graph Neural Network". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

Chung, Fan R. K. (1997). *Spectral Graph Theory*. Conference Board of the Mathematical Sciences.

Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest (1989). *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company.

Corso, Gabriele, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic (2020). "Principal Neighbourhood Aggregation for Graph Nets". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Cortes, Corinna and Vladimir Vapnik (1995). "Support-Vector Networks". In: *Mach. Learn.*, pp. 273–297. DOI: 10.1007/BF00994018.

Costa, Fabrizio and Kurt De Grave (2010). "Fast Neighborhood Subgraph Pairwise Distance Kernel". In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pp. 255–262.

Dai, Hanjun, Zornitsa Kozareva, Bo Dai, Alexander J. Smola, and Le Song (2018). "Learning Steady-States of Iterative Algorithms over Graphs". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 1114–1122.

Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst (2016). "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3837–3845.

Duchi, John C., Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra (2008). "Efficient projections onto the l_1-ball for learning in high dimensions". In: *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pp. 272–279. DOI: 10.1145/1390156.1390191.

Dudley, R. M. (2002). *Real Analysis and Probability*. 2nd ed. Cambridge University Press.

Dwivedi, Vijay Prakash, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson (2023). “Benchmarking Graph Neural Networks”. In: *J. Mach. Learn. Res.*, 43:1–43:48.

Dwivedi, Vijay Prakash, Ladislav Rampásek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini (2022). “Long Range Graph Benchmark”. In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022.*

Eliasof, Moshe, Eldad Haber, and Eran Treister (2024). “Feature Transportation Improves Graph Neural Networks”. In: *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 11874–11882. DOI: `10.1609/AAAI.V38I11.29073`.

Eliasof, Moshe, Lars Ruthotto, and Eran Treister (2023). “Improving Graph Neural Networks with Learnable Propagation Operators”. In: *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, pp. 9224–9245.

Elman, Jeffrey L. (1990). “Finding Structure in Time”. In: *Cogn. Sci.*, pp. 179–211. DOI: `10.1207/S15516709COG1402\_1`.

Erdős, P and A Rényi (1959). “On Random Graphs I”. In: *Publicationes Mathematicae Debrecen*, pp. 290–297.

Feragen, Aasa, Niklas Kasenburg, Jens Petersen, Marleen de Bruijne, and Karsten M. Borgwardt (2013). “Scalable kernels for graphs with continuous attributes”. In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pp. 216–224.

Fey, Matthias (2019). “Just Jump: Dynamic Neighborhood Aggregation in Graph Neural Networks”. In: *CoRR.*

Fey, Matthias, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec (2024). “Position: Relational Deep Learning - Graph Representation Learning on Relational Databases”. In: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024.*

Fey, Matthias and Jan Eric Lenssen (2019). “Fast Graph Representation Learning with PyTorch Geometric”. In: *CoRR.*

Fiedler, Miroslav (1973). “Algebraic connectivity of graphs”. In: *Czechoslovak Mathematical Journal*, pp. 298–305.

Finkelshtein, Ben, Xingyue Huang, Michael M. Bronstein, and İsmail İlkan Ceylan (2024). “Cooperative Graph Neural Networks”. In: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024.*

Freitas, Scott, Yuxiao Dong, Joshua Neil, and Duen Horng Chau (2021). “A Large-Scale Database for Graph Representation Learning”. In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual.*

Fröhlich, Holger, Jörg K. Wegner, Florian Sieker, and Andreas Zell (2005). “Optimal assignment kernels for attributed molecular graphs”. In: *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, pp. 225–232. DOI: 10.1145/1102351.1102380.

Fukushima, Kunihiko (Apr. 1980). “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological Cybernetics*, pp. 193–202. DOI: 10.1007/BF00344251.

Gallager, Robert G. (1996). *Discrete Stochastic Processes.* 1st ed. Springer. DOI: 10.1007/978-1-4615-2329-1.

Gama, Fernando, Antonio G. Marques, Alejandro Ribeiro, and Geert Leus (2018). “MIMO Graph Filters for Convolutional Neural Networks”. In: *19th IEEE International Workshop on Signal Processing Advances in Wireless Communications, SPAWC 2018, Kalamata, Greece, June 25-28, 2018*, pp. 1–5. DOI: 10.1109/SPAWC.2018.8445934.

Gao, Jun, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu (2019). “Representation Degeneration Problem in Training Natural Language Generation Models”. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019.*

Garey, M. R. and David S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman.

Gärtner, Thomas, Peter A. Flach, and Stefan Wrobel (2003). “On Graph Kernels: Hardness Results and Efficient Alternatives”. In: *Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings*, pp. 129–143. DOI: 10.1007/978-3-540-45167-9_11.

Ghaoui, Laurent El, Fangda Gu, Bertrand Travacca, Armin Askari, and Alicia Y. Tsai (2021). “Implicit Deep Learning”. In: *SIAM J. Math. Data Sci.*, pp. 930–958. DOI: 10.1137/20M1358517.

Gilmer, Justin, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl (2017). “Neural Message Passing for Quantum Chemistry”. In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 1263–1272.

Giovanni, Francesco Di, James Rowbottom, Benjamin Paul Chamberlain, Thomas Markovich, and Michael M. Bronstein (2023). “Understanding convolution on graphs via energies”. In: *Trans. Mach. Learn. Res.*

Giunchiglia, Valentina, Chirag Varun Shukla, Guadalupe Gonzalez, and Chirag Agarwal (2022). "Towards Training GNNs Using Explanation Directed Message Passing". In: *Learning on Graphs Conference, LoG 2022, 9-12 December 2022, Virtual Event*, p. 28.

Goldstein, AA (1964). "Convex programming in Hilbert space". In: *Bulletin of the American Mathematical Society*, pp. 709–710.

Goodfellow, Ian J., Yoshua Bengio, and Aaron C. Courville (2016). *Deep Learning*. MIT Press.

Griffiths, Phillip and Joseph Harris (1994). *Principles of algebraic geometry*. Reprint of the 1978 original. John Wiley & Sons, Inc., New York, pp. xiv+813. DOI: 10.1002/9781118032527.

Gu, Fangda, Heng Chang, Wenwu Zhu, Somayeh Sojoudi, and Laurent El Ghaoui (2020). "Implicit Graph Neural Networks". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Guo, Jingwei, Kaizhu Huang, Rui Zhang, and Xinping Yi (2024). "ES-GNN: Generalizing Graph Neural Networks Beyond Homophily With Edge Splitting". In: *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 11345–11360. DOI: 10.1109/TPAMI.2024.3459932.

Guo, Xiaojun, Yifei Wang, Tianqi Du, and Yisen Wang (2023). "ContraNorm: A Contrastive Learning Perspective on Oversmoothing and Beyond". In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.

Hamilton, William L., Zhitao Ying, and Jure Leskovec (2017). "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 1024–1034.

Hammond, David K., Pierre Vandergheynst, and Rémi Gribonval (2011). "Wavelets on graphs via spectral graph theory". In: *Applied and Computational Harmonic Analysis*, pp. 129–150. DOI: https://doi.org/10.1016/j.acha.2010.04.005.

Hansen, Jakob and Thomas Gebhart (2020). "Sheaf Neural Networks". In: *CoRR*.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

He, Mingguo, Zhewei Wei, Zengfeng Huang, and Hongteng Xu (2021). "BernNet: Learning Arbitrary Graph Spectral Filters via Bernstein Approximation". In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 14239–14251.

Henaff, Mikael, Joan Bruna, and Yann LeCun (2015). "Deep Convolutional Networks on Graph-Structured Data". In: *CoRR*.

Hermansson, Linus, Fredrik D. Johansson, and Osamu Watanabe (2015). "Generalized Shortest Path Kernel on Graphs". In: *Discovery Science - 18th International Conference, DS 2015, Banff, AB, Canada, October 4-6, 2015, Proceedings*, pp. 78–85. DOI: 10.1007/978-3-319-24282-8_8.

Hido, Shohei and Hisashi Kashima (2009). "A Linear-Time Graph Kernel". In: *ICDM 2009, The Ninth IEEE International Conference on Data Mining, Miami, Florida, USA, 6-9 December 2009*, pp. 179–188. DOI: 10.1109/ICDM.2009.30.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Comput.*, pp. 1735–1780. DOI: 10.1162/NECO.1997.9.8.1735.

Hogan, Aidan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann (2021). *Knowledge Graphs*. Morgan & Claypool Publishers. DOI: 10.2200/S01125ED1V01Y202109DSK022.

Horn, Roger A. and Charles R. Johnson (1991). *Topics in Matrix Analysis*. Cambridge University Press. DOI: 10.1017/CBO9780511840371.

Hornik, Kurt, Maxwell B. Stinchcombe, and Halbert White (1989). "Multilayer feedforward networks are universal approximators". In: *Neural Networks*, pp. 359–366. DOI: 10.1016/0893-6080(89)90020-8.

Horváth, Tamás, Thomas Gärtner, and Stefan Wrobel (2004). "Cyclic pattern kernels for predictive graph mining". In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pp. 158–167. DOI: 10.1145/1014052.1014072.

Hu, Weihua, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec (2021). "OGB-LSC: A Large-Scale Challenge for Machine Learning on Graphs". In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 448–456.

Jamadandi, Adarsh, Celia Rubio-Madrigal, and Rebekka Burkholz (2024). "Spectral Graph Pruning Against Over-Squashing and Over-Smoothing". In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Jin, Wei, Xiaorui Liu, Yao Ma, Charu C. Aggarwal, and Jiliang Tang (2022). "Feature Overcorrelation in Deep Graph Neural Networks: A New Perspective". In: *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pp. 709–719. DOI: 10.1145/3534678.3539445.

Jing, Li, Pascal Vincent, Yann LeCun, and Yuandong Tian (2022). "Understanding Dimensional Collapse in Contrastive Self-supervised Learning". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022.*

Johansson, Fredrik D. and Devdatt P. Dubhashi (2015). "Learning with Similarity Functions on Graphs using Matchings of Geometric Embeddings". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pp. 467–476. DOI: 10.1145/2783258.2783341.

Johansson, Fredrik D., Vinay Jethava, Devdatt P. Dubhashi, and Chiranjib Bhattacharyya (2014). "Global graph kernels using geometric embeddings". In: *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 694–702.

Kashima, Hisashi, Koji Tsuda, and Akihiro Inokuchi (2003). "Marginalized Kernels Between Labeled Graphs". In: *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pp. 321–328.

Keriven, Nicolas (2022). "Not too little, not too much: a theoretical analysis of graph (over)smoothing". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022.*

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.*

Kipf, Thomas N. and Max Welling (2017). "Semi-Supervised Classification with Graph Convolutional Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings.*

Klicpera, Johannes, Aleksandar Bojchevski, and Stephan Günnemann (2019). "Predict then Propagate: Graph Neural Networks meet Personalized PageRank". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019.*

Knabner, Peter and Wolf Barth (2017). "Lineare Algebra und Analysis". In: *Lineare Algebra: Aufgaben und Lösungen*, pp. 229–242. DOI: 10.1007/978-3-662-54991-9_15.

Kothapalli, Vignesh, Tom Tirer, and Joan Bruna (2023). "A Neural Collapse Perspective on Feature Evolution in Graph Neural Networks". In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023.*

Kowalewski, Gerhard (1909). *Einführung in die Determinantentheorie*. De Gruyter. DOI: doi:10.1515/9783112342442.

Kreuzer, Devin, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou (2021). "Rethinking Graph Transformers with Spectral Attention". In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 21618–21629.

Kriege, Nils M., Pierre-Louis Giscard, Franka Bause, and Richard C. Wilson (2019). "Computing Optimal Assignments in Linear Time for Approximate Graph Matching". In: *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*, pp. 349–358. DOI: 10.1109/ICDM.2019.00045.

Kriege, Nils M., Pierre-Louis Giscard, and Richard C. Wilson (2016). "On Valid Optimal Assignment Kernels and Applications to Graph Classification". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 1615–1623.

Kriege, Nils M., Fredrik D. Johansson, and Christopher Morris (2020). "A survey on graph kernels". In: *Appl. Netw. Sci.*, p. 6. DOI: 10.1007/S41109-019-0195-3.

Kriege, Nils M., Christopher Morris, Anja Rey, and Christian Sohler (2018). "A Property Testing Framework for the Theoretical Expressivity of Graph Kernels". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pp. 2348–2354. DOI: 10.24963/IJCAI.2018/325.

Kriege, Nils M. and Petra Mutzel (2012). "Subgraph Matching Kernels for Attributed Graphs". In: *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*.

Kriege, Nils M., Marion Neumann, Kristian Kersting, and Petra Mutzel (2014). "Explicit Versus Implicit Graph Feature Maps: A Computational Phase Transition for Walk Kernels". In: *2014 IEEE International Conference on Data Mining, ICDM 2014, Shenzhen, China, December 14-17, 2014*, pp. 881–886. DOI: 10.1109/ICDM.2014.129.

Kriege, Nils M., Marion Neumann, Christopher Morris, Kristian Kersting, and Petra Mutzel (2019). "A unifying view of explicit and implicit feature maps of graph kernels". In: *Data Min. Knowl. Discov.*, pp. 1505–1547. DOI: 10.1007/S10618-019-00652-0.

Lachi, Veronica, Alice Moallemy-Oureh, Andreas Roth, and Pascal Welke (2025). "Expressive Pooling for Graph Neural Networks". In: *Transactions on Machine Learning Research*.

LeCun, Yann, Yoshua Bengio, and Geoffrey E. Hinton (2015). "Deep learning". In: *Nat.*, pp. 436–444. DOI: 10.1038/NATURE14539.

LeCun, Yann, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel (1989). "Backpropagation Applied to Hand-written Zip Code Recognition". In: *Neural Comput.*, pp. 541–551. DOI: 10.1162/NECO.1989.1.4.541.

Li, Guohao, Matthias Müller, Bernard Ghanem, and Vladlen Koltun (2021). "Training Graph Neural Networks with 1000 Layers". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pp. 6437–6449.

Li, Guohao, Chenxin Xiong, Ali K. Thabet, and Bernard Ghanem (2020). "DeeperGCN: All You Need to Train Deeper GCNs". In: *CoRR*.

Li, Qimai, Zhichao Han, and Xiao-Ming Wu (2018). "Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3538–3545. DOI: 10.1609/AAAI.V32I1.11604.

Li, Yujia, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel (2016). "Gated Graph Sequence Neural Networks". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Lim, Derek, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser-Nam Lim (2021). "Large Scale Learning on Non-Homophilous Graphs: New Benchmarks and Strong Simple Methods". In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 20887–20902.

Liu, Meng, Hongyang Gao, and Shuiwang Ji (2020). "Towards Deeper Graph Neural Networks". In: *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pp. 338–348. DOI: 10.1145/3394486.3403076.

Loshchilov, Ilya and Frank Hutter (2019). "Decoupled Weight Decay Regularization". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Luan, Sitao, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup (2022). "Revisiting Heterophily For Graph Neural Networks". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Luks, Eugene M. (1982). "Isomorphism of Graphs of Bounded Valence can be Tested in Polynomial Time". In: *J. Comput. Syst. Sci.*, pp. 42–65. DOI: 10.1016/0022-0000(82)90009-5.

Luxburg, Ulrike von (2007). "A tutorial on spectral clustering". In: *Stat. Comput.*, pp. 395–416. DOI: 10.1007/S11222-007-9033-Z.

Maskey, Sohir, Raffaele Paolino, Aras Bacho, and Gitta Kutyniok (2023). “A Fractional Graph Laplacian Approach to Oversmoothing”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

McCallum, Andrew Kachites, Kamal Nigam, Jason Rennie, and Kristie Seymore (2000). “Automating the Construction of Internet Portals with Machine Learning”. In: *Inf. Retr.*, pp. 127–163. DOI: 10.1023/A:1009953814988.

Mises, R. V. and H. Pollaczek-Geiringer (1929). “Praktische Verfahren der Gleichungsauflösung .” In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, pp. 152–164. DOI: https://doi.org/10.1002/zamm.19290090206.

Morris, Christopher, Kristian Kersting, and Petra Mutzel (2017). “Glocalized Weisfeiler-Lehman Graph Kernels: Global-Local Feature Maps of Graphs”. In: *2017 IEEE International Conference on Data Mining, ICDM 2017, New Orleans, LA, USA, November 18-21, 2017*, pp. 327–336. DOI: 10.1109/ICDM.2017.42.

Morris, Christopher, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann (2020). “TUDataset: A collection of benchmark datasets for learning with graphs”. In: *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*.

Morris, Christopher, Nils M. Kriege, Kristian Kersting, and Petra Mutzel (2016). “Faster Kernels for Graphs with Continuous Attributes via Hashing”. In: *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pp. 1095–1100. DOI: 10.1109/ICDM.2016.0142.

Morris, Christopher, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe (2019). “Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks”. In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 4602–4609. DOI: 10.1609/AAAI.V33I01.33014602.

Nguyen, Khang, Nong Minh Hieu, Vinh Duc Nguyen, Nhat Ho, Stanley J. Osher, and Tan Minh Nguyen (2023). “Revisiting Over-smoothing and Over-squashing Using Ollivier-Ricci Curvature”. In: *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, pp. 25956–25979.

NT, Hoang and Takanori Maehara (2019). “Revisiting Graph Neural Networks: All We Have is Low-Pass Filters”. In: *CoRR*.

O’Neil, Richard (1963). “Convolution operators and l(p,q) spaces”. In: *Duke Mathematical Journal* (1). DOI: 10.1215/s0012-7094-63-03015-1.

Oneto, Luca, Nicolò Navarin, Michele Donini, Alessandro Sperduti, Fabio Aiolli, and Davide Anguita (2017). "Measuring the expressivity of graph kernels through Statistical Learning Theory". In: *Neurocomputing*, pp. 4–16. DOI: 10.1016/J.NEUCOM.2017.02.088.

Oono, Kenta and Taiji Suzuki (2020). "Graph Neural Networks Exponentially Lose Expressive Power for Node Classification". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.*

OpenAI (2023). "GPT-4 Technical Report". In: *CoRR*. DOI: 10.48550/ARXIV.2303.08774.

Oppenheim, Alan and Ronald Schafer (2013). *Discrete-Time Signal Processing*. Pearson Deutschland, p. 1056.

Orsini, Francesco, Paolo Frasconi, and Luc De Raedt (2015). "Graph Invariant Kernels". In: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 3756–3762.

Page, Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd (Nov. 1999). *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report. Previous number = SIDL-WP-1999-0120. Stanford InfoLab.

Pascanu, Razvan, Tomás Mikolov, and Yoshua Bengio (2013). "On the difficulty of training recurrent neural networks". In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pp. 1310–1318.

Pei, Hongbin, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang (2020). "Geom-GCN: Geometric Graph Convolutional Networks". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.*

Pérez-Villanueva, Jaime, Oscar Méndez-Lucio, Olivia Soria-Arteche, and José L. Medina-Franco (Nov. 2015). "Activity cliffs and activity cliff generators based on chemotype-related activity landscapes". In: *Molecular Diversity*, pp. 1021–1035. DOI: 10.1007/s11030-015-9609-z.

Platonov, Oleg, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova (2023). "A critical look at the evaluation of GNNs under heterophily: Are we really making progress?" In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023.*

Rampásek, Ladislav, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini (2022). "Recipe for a General, Powerful, Scalable Graph Transformer". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022.*

Rampásek, Ladislav and Guy Wolf (2021). "Hierarchical Graph Neural Nets can Capture Long-Range Interactions". In: *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP), Gold Coast, Australia, October 25-28, 2021*, pp. 1–6. DOI: 10.1109/MLSP52302.2021.9596069.

Ribeiro, Leonardo Filipe Rodrigues, Pedro H. P. Saverese, and Daniel R. Figueiredo (2017). "*struc2vec*: Learning Node Representations from Structural Identity". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pp. 385–394. DOI: 10.1145/3097983.3098061.

Rong, Yu, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang (2020). "Self-Supervised Graph Transformer on Large-Scale Molecular Data". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

Rong, Yu, Wenbing Huang, Tingyang Xu, and Junzhou Huang (2020). "DropEdge: Towards Deep Graph Convolutional Networks on Node Classification". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.*

Rosenbluth, Eran, Jan Tönshoff, and Martin Grohe (2023). "Some Might Say All You Need Is Sum". In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pp. 4172–4179. DOI: 10.24963/IJCAI.2023/464.

Rossi, Emanuele, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael M. Bronstein (2023). "Edge Directionality Improves Learning on Heterophilic Graphs". In: *Learning on Graphs Conference, 27-30 November 2023, Virtual Event*, p. 25.

Roth, Andreas (2024). "Simplifying the Theory on Over-Smoothing". In: *CoRR.* DOI: 10.48550/ARXIV.2407.11876.

Roth, Andreas, Franka Bause, Nils Morten Kriege, and Thomas Liebig (2024). "Preventing Representational Rank Collapse in MPNNs by Splitting the Computational Graph". In: *Learning on Graphs Conference, 26-29 November 2024, Virtual*, p. 14.

Roth, Andreas and Thomas Liebig (2022a). "Forecasting Unobserved Node States with spatio-temporal Graph Neural Networks". In: *IEEE International Conference on Data Mining Workshops, ICDM 2022 - Workshops, Orlando, FL, USA, November 28 - Dec. 1, 2022*, pp. 740–747. DOI: 10.1109/ICDMW58026.2022.00101.

Roth, Andreas and Thomas Liebig (2022b). "Transforming PageRank into an Infinite-Depth Graph Neural Network". In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2022, Grenoble, France, September 19-23, 2022, Proceedings, Part II*, pp. 469–484. DOI: 10.1007/978-3-031-26390-3_27.

Roth, Andreas and Thomas Liebig (2023). "Rank Collapse Causes Over-Smoothing and Over-Correlation in Graph Neural Networks". In: *Learning on Graphs Conference, 27-30 November 2023, Virtual Event*, p. 35.

Roth, Andreas and Thomas Liebig (2025). "What Can We Learn From MIMO Graph Convolutions?" In: *Proceedings of the Thirty-Fourth International Joint Conference on Artifi-*

cial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025, pp. 6120–6128. DOI: 10.24963/IJCAI.2025/681.

Roth, Andreas, Konstantin Wüstefeld, and Frank Weichert (2021). "A Data-Centric Augmentation Approach for Disturbed Sensor Image Segmentation". In: *J. Imaging*, p. 206. DOI: 10.3390/JIMAGING7100206.

Rozemberczki, Benedek, Carl Allen, and Rik Sarkar (2021). "Multi-Scale attributed node embedding". In: *J. Complex Networks.* DOI: 10.1093/COMNET/CNAB014.

Rubio-Madrigal, Celia, Adarsh Jamadandi, and Rebekka Burkholz (2025). "GNNs Getting ComFy: Community and Feature Similarity Guided Rewiring". In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025.*

Rusch, T. Konstantin, Michael M. Bronstein, and Siddhartha Mishra (2023). "A Survey on Oversmoothing in Graph Neural Networks". In: *CoRR.* DOI: 10.48550/ARXIV.2303.10993.

Rusch, T. Konstantin, Ben Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael M. Bronstein (2022). "Graph-Coupled Oscillator Networks". In: *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, pp. 18888–18909.

Rusch, T. Konstantin, Benjamin Paul Chamberlain, Michael W. Mahoney, Michael M. Bronstein, and Siddhartha Mishra (2023). "Gradient Gating for Deep Multi-Rate Learning on Graphs". In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023.*

Sanders, Cedric, Andreas Roth, and Thomas Liebig (2023). "Curvature-Based Pooling Within Graph Neural Networks". In: *Machine Learning and Principles and Practice of Knowledge Discovery in Databases - International Workshops of ECML PKDD 2023, Turin, Italy, September 18-22, 2023, Revised Selected Papers, Part III*, pp. 471–485. DOI: 10.1007/978-3-031-74633-8_35.

Sandryhaila, Aliaksei and José M. F. Moura (2013). "Discrete Signal Processing on Graphs". In: *IEEE Trans. Signal Process.*, pp. 1644–1656. DOI: 10.1109/TSP.2013.2238935.

Schlichtkrull, Michael Sejr, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling (2018). "Modeling Relational Data with Graph Convolutional Networks". In: *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, pp. 593–607. DOI: 10.1007/978-3-319-93417-4_38.

Schmidhuber, Jürgen (2015). "Deep learning in neural networks: An overview". In: *Neural Networks*, pp. 85–117. DOI: 10.1016/J.NEUNET.2014.09.003.

Scholkemper, Michael, Xinyi Wu, Ali Jadbabaie, and Michael T. Schaub (2025). "Residual Connections and Normalization Can Provably Prevent Oversmoothing in GNNs". In:

The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025.

Sen, Prithviraj, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad (2008). "Collective Classification in Network Data". In: *AI Mag.*, pp. 93–106. DOI: 10.1609/AIMAG.V29I3.2157.

Shervashidze, Nino, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt (2011). "Weisfeiler-Lehman Graph Kernels". In: *J. Mach. Learn. Res.*, pp. 2539–2561. DOI: 10.5555/1953048.2078187.

Shervashidze, Nino, S. V. N. Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten M. Borgwardt (2009). "Efficient graphlet kernels for large graph comparison". In: *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*, pp. 488–495.

Shi, Yunsheng, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun (2021). "Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification". In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pp. 1548–1554. DOI: 10.24963/IJCAI.2021/214.

Shuman, David I., Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst (2013). "The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains". In: *IEEE Signal Process. Mag.*, pp. 83–98. DOI: 10.1109/MSP.2012.2235192.

Stachenfeld, Kimberly L., Jonathan Godwin, and Peter W. Battaglia (2021). "Graph Networks with Spectral Message Passing". In: *CoRR*.

Stärk, Hannes, Octavian Ganea, Lagnajit Pattanaik, Regina Barzilay, and Tommi S. Jaakkola (2022). "EquiBind: Geometric Deep Learning for Drug Binding Structure Prediction". In: *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, pp. 20503–20521.

Sterling, Teague and John J. Irwin (2015). "ZINC 15 - Ligand Discovery for Everyone". In: *J. Chem. Inf. Model.*, pp. 2324–2337. DOI: 10.1021/ACS.JCIM.5B00559.

Sun, Qingyun, Jianxin Li, Haonan Yuan, Xingcheng Fu, Hao Peng, Cheng Ji, Qian Li, and Philip S. Yu (2022). "Position-aware Structure Learning for Graph Topology-imbalance by Relieving Under-reaching and Over-squashing". In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, pp. 1848–1857. DOI: 10.1145/3511808.3557419.

Suresh, Susheel, Vinith Budde, Jennifer Neville, Pan Li, and Jianzhu Ma (2021). "Breaking the Limit of Graph Neural Networks by Improving the Assortativity of Graphs with Local Mixing Patterns". In: *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pp. 1541–1551. DOI: 10.1145/3447548.3467373.

Tailor, Shyam A., Felix L. Opolka, Pietro Liò, and Nicholas Donald Lane (2022). "Do We Need Anisotropic Graph Neural Networks?" In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022.*

Tang, Jie, Jimeng Sun, Chi Wang, and Zi Yang (2009). "Social influence analysis in large-scale networks". In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pp. 807–816. DOI: 10.1145/1557019.1557108.

Tao, Terence (2012). *Topics in random matrix theory.* American Mathematical Soc.

Tönshoff, Jan, Martin Ritzert, Eran Rosenbluth, and Martin Grohe (2024). "Where Did the Gap Go? Reassessing the Long-Range Graph Benchmark". In: *Trans. Mach. Learn. Res.*

Topping, Jake, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein (2022). "Understanding over-squashing and bottlenecks on graphs via curvature". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022.*

Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample (2023). "LLaMA: Open and Efficient Foundation Language Models". In: *CoRR.* DOI: 10.48550/ARXIV.2302.13971.

Vashishth, Shikhar, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar (2020). "Composition-based Multi-Relational Graph Convolutional Networks". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.*

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008.

Velickovic, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio (2018). "Graph Attention Networks". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings.*

Vishwanathan, S. V. N., Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt (2010). "Graph Kernels". In: *J. Mach. Learn. Res.*, pp. 1201–1242. DOI: 10.5555/1756006.1859891.

Weisfeiler, Boris and A. A. Lehman (1968). "A Reduction of a Graph to a Canonical Form and an Algebra Arising During This Reduction". In: *Nauchno-Technicheskaya Informatsia*, pp. 12–16.

Wells, Alexander Frank (2012). *Structural inorganic chemistry*. Oxford University Press, USA.

Woznica, Adam, Alexandros Kalousis, and Melanie Hilario (2010). "Adaptive Matching Based Kernels for Labelled Graphs". In: *Advances in Knowledge Discovery and Data Mining, 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, June 21-24, 2010. Proceedings. Part II*, pp. 374–385. DOI: 10.1007/978-3-642-13672-6_37.

Wu, Xinyi, Amir Ajorlou, Zihui Wu, and Ali Jadbabaie (2023). "Demystifying Oversmoothing in Attention-Based Graph Neural Networks". In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Xu, Keyulu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka (2019). "How Powerful are Graph Neural Networks?" In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Xu, Keyulu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka (2018). "Representation Learning on Graphs with Jumping Knowledge Networks". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 5449–5458.

Yan, Yujun, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra (2022). "Two Sides of the Same Coin: Heterophily and Oversmoothing in Graph Convolutional Neural Networks". In: *IEEE International Conference on Data Mining, ICDM 2022, Orlando, FL, USA, November 28 - Dec. 1, 2022*, pp. 1287–1292. DOI: 10.1109/ICDM54844.2022.00169.

Yanardag, Pinar and S. V. N. Vishwanathan (2015). "Deep Graph Kernels". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pp. 1365–1374. DOI: 10.1145/2783258.2783417.

Yang, Jaewon and Jure Leskovec (2012). "Defining and Evaluating Network Communities Based on Ground-Truth". In: *12th IEEE International Conference on Data Mining, ICDM 2012, Brussels, Belgium, December 10-13, 2012*, pp. 745–754. DOI: 10.1109/ICDM.2012.138.

Yang, Jaewon and Jure Leskovec (2015). "Defining and evaluating network communities based on ground-truth". In: *Knowl. Inf. Syst.*, pp. 181–213. DOI: 10.1007/S10115-013-0693-Z.

Yang, Yiding, Zunlei Feng, Mingli Song, and Xinchao Wang (2020). "Factorizable Graph Convolutional Networks". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Yang, Zhilin, William W. Cohen, and Ruslan Salakhutdinov (2016). "Revisiting Semi-Supervised Learning with Graph Embeddings". In: *Proceedings of the 33nd International*

Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016, pp. 40–48.

You, Jiaxuan, Tianyu Du, and Jure Leskovec (2022). "ROLAND: Graph Learning Framework for Dynamic Graphs". In: *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pp. 2358–2366. DOI: 10.1145/3534678.3539300.

Zachary, Wayne W (1977). "An information flow model for conflict and fission in small groups". In: *Journal of anthropological research*, pp. 452–473.

Zhang, Muhan, Zhicheng Cui, Marion Neumann, and Yixin Chen (2018). "An End-to-End Deep Learning Architecture for Graph Classification". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 4438–4445. DOI: 10.1609/AAAI.V32I1.11782.

Zhao, Lingxiao and Leman Akoglu (2020). "PairNorm: Tackling Oversmoothing in GNNs". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Zhou, Kaixiong, Xiao Huang, Daochen Zha, Rui Chen, Li Li, Soo-Hyun Choi, and Xia Hu (2021). "Dirichlet Energy Constrained Learning for Deep Graph Neural Networks". In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 21834–21846.

Zitnik, Marinka and Jure Leskovec (2017). "Predicting multicellular function through multi-layer tissue networks". In: *Bioinform.*, pp. i190–i198. DOI: 10.1093/BIOINFORMATICS/BTX252.