What Can We Learn From MIMO Graph Convolutions?

Andreas Roth[1](✉)[0000−0002−0515−7635] and Thomas Liebig[1][0000−0002−9841−1101]

TU Dortmund University, Dortmund, Germany
{andreas.roth,thomas.liebig}@tu-dortmund.de

Abstract. Most graph neural networks (GNNs) utilize approximations of the general graph convolution derived in the graph Fourier domain. While GNNs are typically applied in the multi-input multi-output (MIMO) case, the approximations are performed in the single-input single-output (SISO) case. In this work, we first derive the MIMO graph convolution through the convolution theorem and approximate it directly in the MIMO case. We find the key MIMO-specific property of the graph convolution to be operating on multiple computational graphs, or equivalently, applying distinct feature transformations for each pair of nodes. As a localized approximation, we introduce localized MIMO graph convolutions (LMGCs), which generalize many linear message-passing neural networks. For almost every choice of edge weights, we prove that LMGCs with a single computational graph are injective on multisets, and the resulting representations are linearly independent when more than one computational graph is used. Our experimental results confirm that an LMGC can combine the benefits of various methods.

Keywords: graph machine learning · geometric deep learning · graph convolutions.

1 Introduction

Graph neural networks have emerged as an effective method for many challenging applications involving graph-structured data, e.g., molecular prediction [14]. These utilize convolutional operations typically derived from the general graph convolution obtained in the Fourier domain, as given by the convolution theorem [11,4]. Initially, approximations of the general graph convolution were based on polynomials, e.g., Chebyshev polynomials [11]. The graph convolutional network (GCN) [16] approximates these polynomials as a first-order localization. Many other message-passing approaches are derived from the GCN [37,1]. However, these approximations are based on the single-output single-input (SISO) case, where the input and output contain a single feature for each node. GNNs are typically applied in the multi-input multi-output (MIMO) case, where each node has multiple feature channels assigned, and the output also contains multiple features. Extending from the SISO to the MIMO case is achieved by applying these methods for each input and output channel combination and learning distinct parameters [4,6,16].

Instead of first approximating the graph convolution in the SISO and then extending to the MIMO, we propose directly performing the approximation in the MIMO case to

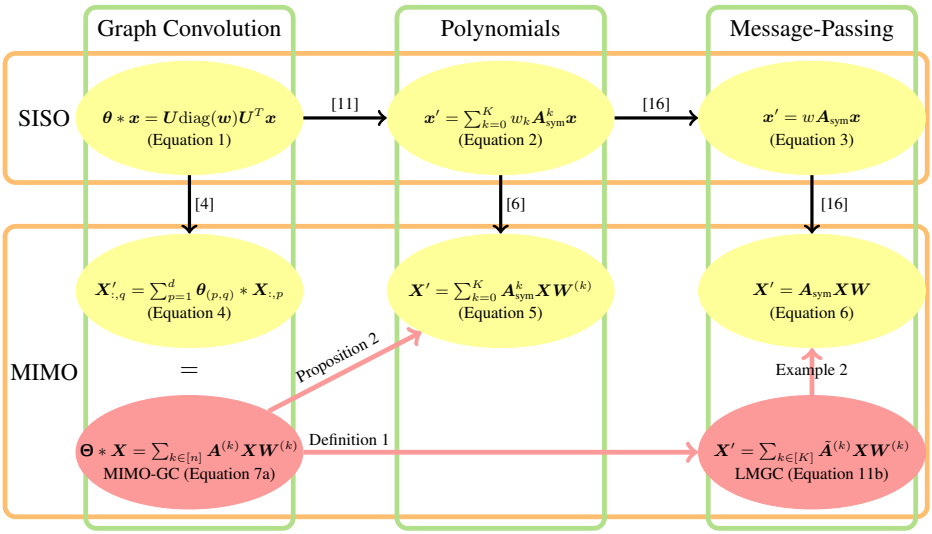


Fig. 1: Connections between the graph convolution, polynomial filters, and message-passing approaches in the SISO and the MIMO case. Parts in yellow (■) indicate existing contributions, parts in pink (■) our contributions.

benefit from MIMO-specific properties. We first derive the general graph convolution in the MIMO case through the convolution theorem and the graph Fourier transform. We find the key property that allows the MIMO-GC to represent arbitrary transformations to be operating on multiple computational graphs or, equivalently, applying distinct linear feature transformations between each pair of nodes. This form allows a direct approximation in the MIMO case by localizing the aggregation step. The resulting localized MIMO-GC (LMGC) presents a general framework for linear message-passing neural networks (MPNN) that inherits the beneficial properties for multi-channel learning. While we show that the LMGC can represent most MPNNs, the LMGC cannot represent the graph isomorphism network (GIN) [39] due to its non-linear feature transformation. However, we show that LMGCs are injective on multisets for almost every choice of edge weights even for a single computational graph. When further utilizing multiple computational graphs as motivated by the MIMO-GC, we prove that representations are linearly independent for almost every choice of edge weights. We summarize our main contributions as follows:

- Based on the convolution theorem, we derive the MIMO graph convolution (MIMO-GC) for node representations with multiple feature channels. A key property of MIMO-GCs is to operate on multiple computational graphs, or equivalently, to apply distinct linear feature transformations for each pair of nodes (Section 3).
- We introduce the framework of localized MIMO-GCs (LMGCs) by localizing the aggregation step of the MIMO-GC. It merges the key idea of operating on multiple computational graphs with the efficient message-passing scheme (Section 4).

– We prove that LMGCs are injective on multisets for a single computational graph and produce linearly independent representations when more than one computational graph is used for almost every choice of edge weights (Section 4).

2 Preliminaries

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a connected and undirected graph consisting of a set of n nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$. Let $\boldsymbol{A} \in \{0,1\}^{n \times n}$ be the corresponding adjacency matrix with $A_{i,j} = 1$ if $(i,j) \in \mathcal{E}$ and 0 otherwise. The diagonal degree matrix is $\boldsymbol{D} \in \mathbb{N}^{n \times n}$. The symmetrically normalized adjacency matrix is given by $\boldsymbol{A}_{\text{sym}} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ and the graph Laplacian by $\boldsymbol{L}_{\text{sym}} = \boldsymbol{I}_n - \boldsymbol{A}_{\text{sym}}$. Its eigendecomposition is $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ where $\boldsymbol{\Lambda} \in \mathbb{R}^{n \times n}$ is a diagonal matrix containing its eigenvalues, and $\boldsymbol{U} \in \mathbb{R}^{n \times n}$ is an orthonormal matrix containing the corresponding eigenvectors as columns. We refer to a vector $\boldsymbol{x} \in \mathbb{R}^n$ as a single-channel graph signal and to a matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ as a multi-channel graph signal. These can be initial features or expressive and informative node embeddings. In the graph domain, the Fourier base is given by the eigenvectors $\boldsymbol{U}^T$ of the graph Laplacian. Thus, the Fourier transformation $F = \boldsymbol{U}^T$ is performed by projecting a graph signal onto the eigenvectors, and its inverse transformation is given by $F^{-1} = \boldsymbol{U}$. We further refer to $\boldsymbol{U}_{i,:}\boldsymbol{x} \in \mathbb{R}$ as the component of $\boldsymbol{x}$ corresponding to vector $\boldsymbol{U}_{i,:}$.

2.1 Graph Convolutions

Given a graph signal, the graph convolution or a similar method derived from it are designed to obtain a more informative graph signal. In the SISO case, the input and output are single-channel graph signals, while in the MIMO case, they are multi-channel graph signals. Graph neural networks (GNNs) are typically constructed by interleaving these operations with non-linear activation functions. The following derivations and approximations of the graph convolution are visualized in Fig. 1.

The general graph convolution is defined in the SISO case through the convolution theorem using the graph Fourier transform as

$$\boldsymbol{\theta} * \boldsymbol{x} = \boldsymbol{U}\text{diag}(\boldsymbol{w})\boldsymbol{U}^T\boldsymbol{x} \tag{1}$$

where $\boldsymbol{w} = \boldsymbol{U}^T\boldsymbol{\theta} \in \mathbb{R}^n$ [11,4]. As $\boldsymbol{x}$ and $\boldsymbol{\theta} * \boldsymbol{x}$ are single-channel signals, we will refer to this as the SISO graph convolution (SISO-GC).

Due to the runtime and memory complexity and inability to apply the same graph Fourier transform across graphs, most GNNs utilize approximations of the SISO-GC. Polynomials in $\boldsymbol{A}_{\text{sym}}$ (or equivalently $\boldsymbol{L}_{\text{sym}}$) provide a K-localized approximation

$$\boldsymbol{\theta} * \boldsymbol{x} \approx \sum_{k=0}^{K} w_{(k)} \boldsymbol{A}_{\text{sym}}^k \boldsymbol{x} \tag{2}$$

of the SISO-GC where $w_k \in \mathbb{R}$ are scalars for $k \in [K]$ [11]. Examples of such approximations are Chebyshev [11] and Cayley polynomials [20].

Similarly, the graph convolutional network (GCN) [16] was derived as a first-order localization

$$\boldsymbol{\theta} * \boldsymbol{x} \approx w\boldsymbol{A}_{\text{sym}}\boldsymbol{x} \tag{3}$$

of SISO polynomials using a single parameter $w \in \mathbb{R}$.

The graph convolution has not yet been derived for the MIMO case. Instead, following Bruna et al. [4], the graph convolution and the described approximations are extended to the MIMO case by applying it to each combination of input channel $p \in [d]$ and output channel $q \in [c]$. For the graph convolution, the output

$$\boldsymbol{X}'_{:,q} = \sum_{p=1}^{d} \boldsymbol{\theta}_{(p,q)} * \boldsymbol{X}_{:,p} \tag{4}$$

is obtained by defining distinct filters $\theta_{(p,q)} \in \mathbb{R}^n$.

SISO polynomials are equivalently extended to the MIMO case by applying distinct parameters $W^{(k)}_{p,q} \in \mathbb{R}$ for each combination of input and output channels [6]. Based on Equation 2, we have

$$\boldsymbol{X}'_{:,q} = \sum_{p=1}^{d} \sum_{k=0}^{K} W^{(k)}_{p,q} \boldsymbol{A}^k_{\text{sym}} \boldsymbol{X}_{:,p} \tag{5}$$

where $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d \times c}$.

Equivalently, the GCN is applied in the MIMO case using distinct parameters $W_{p,q} \in \mathbb{R}$ for each combination of input channel p and output channel q [16]. This led to the typical form of

$$\boldsymbol{X}'_{:,q} = \sum_{p=1}^{d} W_{p,q} \boldsymbol{A}_{\text{sym}} \boldsymbol{X}_{:,p} = [\boldsymbol{A}_{\text{sym}} \boldsymbol{X} \boldsymbol{W}]_{:,q} \,. \tag{6}$$

Most other message-passing methods were then further derived from the GCN. In this work, we show the advantages of directly obtaining the graph convolution and approximations in the MIMO case.

3 MIMO Graph Convolution

We now consider the MIMO case. Let $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ be a multi-channel graph signal with d channels for each node. The multi-channel output signal $\boldsymbol{Y} \in \mathbb{R}^{n \times c}$ can have a different number of channels c. We first derive the general graph convolution for the MIMO through the convolution theorem [24] and the graph Fourier transform. The filter $\boldsymbol{\Theta} \in \mathbb{R}^{n \times c \times d}$ contains the necessary element-wise mappings from d to c dimensions. To the best of our knowledge, this has not yet been derived.

Theorem 1 (MIMO Graph Convolution (MIMO-GC)). *Let $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, $\boldsymbol{\Theta} \in \mathbb{R}^{n \times c \times d}$, and the Fourier transform $F = \boldsymbol{U}^T \in \mathbb{R}^{n \times n}$ be given by the eigenvectors*

of the graph Laplacian $\boldsymbol{\Lambda}$. Then, their convolution is given as

$$(\boldsymbol{\Theta} * \boldsymbol{X})(i) = \left[\sum_{k=1}^{n} \boldsymbol{A}^{(k)} \boldsymbol{X} \boldsymbol{W}^{(k)}\right]_{i,:} \tag{7a}$$

$$= \sum_{j=1}^{n} \boldsymbol{W}_{(i,j)} \boldsymbol{X}_{j,:} \in \mathbb{R}^{c} \tag{7b}$$

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

We provide all detailed proofs as supplementary material. The MIMO-GC is unique because it does not require additional definitions from us. As such, MIMO operations on graphs should closely approximate the MIMO-GC. We note that the MIMO-GC is equivalent to extending the SISO-GC to multi-channel signals by applying it to every pair of input and output channels, as introduced by Bruna et al. [4]. The MIMO-GC can be interpreted in two ways.

Based on Eq. (7a), each $\boldsymbol{A}^{(k)} \in \mathbb{R}^{n \times n}$ can be seen as a fully connected computational graph with edge weights $A^{(k)}_{i,j} = U_{i,k} \cdot U_{j,k} \in \mathbb{R}$ given by the corresponding Fourier basis vector $\boldsymbol{U}_{:,k}$. This form is also similar to multi-head self-attention [36]. However, they normalize edge weights by the softmax activation, preventing them from being orthogonal across heads. The corresponding parameter matrix $\boldsymbol{W}^{(k)}$ specifies how much this component is amplified or damped from each input channel to each output channel. Utilizing n computational graphs allows the MIMO-GC to amplify distinct components for each output channel. Assuming all components are present in the input signal, the MIMO-GC can produce any output signal:

Proposition 1 (Universality of the MIMO-GC). *For any $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ with $\boldsymbol{U}^T \boldsymbol{X} \neq_{em} 0$ element-wise non-zero and any $\boldsymbol{Y} \in \mathbb{R}^{n \times c}$, there exists a $\boldsymbol{\Theta} \in \mathbb{R}^{n \times c \times d}$, such that*

$$\boldsymbol{\Theta} * \boldsymbol{X} = \boldsymbol{Y} \,. \tag{8}$$

Based on Eq. (7b), the MIMO-GC can also be interpreted as applying distinct feature transformation $\boldsymbol{W}_{(i,j)}$ for each pair of nodes. Each $\boldsymbol{W}_{(i,j)}$ is a unique linear combination of a shared set of n feature transformations. Relatedly, utilizing distinct feature transformations was recently popularized as Neural Sheaf Diffusion [12,2]. The MIMO-GC provides an additional theoretical justification for such methods.

However, computing the MIMO-GC exactly is typically not desirable, as with the SISO-GC. It is inherently transductive, as the graph Fourier transform is graph-dependent, and thus, a learned filter cannot be applied to novel or changed graphs. Most importantly, the computational complexity of the graph convolution scales quadratically with the number of nodes:

Computational Complexity. Equivalently to computing the SISO-GC exactly, the total complexity of the MIMO-GC is dominated by the graph Fourier transform as it requires dense matrix multiplications. The overall complexity is thus $\mathcal{O}(n^2 \cdot c \cdot d)$.

Benefiting from the MIMO-GC. Instead of directly computing the MIMO-GC, we aim to improve the approximations previously derived from the SISO-GC, which were then extended to the MIMO case. We first confirm that these MIMO polynomials are also approximations of the MIMO-GC with constraints on the allowed filters $\boldsymbol{\Theta}$:

Proposition 2 (Every MIMO polynomial filter is a MIMO-GC with a specific filter). *Let $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ for some $d \in \mathbb{N}$. For any $\boldsymbol{V}^{(0)}, \dots, \boldsymbol{V}^{(K)} \in \mathbb{R}^{d \times c}$ with $c, K \in \mathbb{N}$, there exists a $\boldsymbol{\Theta}_{poly} \in \mathbb{R}^{n \times c \times d}$, such that*

$$\sum_{k=0}^{K} \boldsymbol{A}_{sym}^{k} \boldsymbol{X} \boldsymbol{V}^{(k)} = \boldsymbol{\Theta}_{poly} * \boldsymbol{X} \,. \tag{9}$$

As one such example of a first-degree polynomial, the GCN is a MIMO-GC with specific constraints on $\boldsymbol{\Theta}$:

Example 1 (GCN is a MIMO-GC). Let $\boldsymbol{X} \in \mathbb{R}^{n \times d}, \boldsymbol{V} \in \mathbb{R}^{d \times c}$. Then,

$$\begin{aligned} \boldsymbol{A}_{\text{sym}} \boldsymbol{X} \boldsymbol{V} &= \sum_{k=1}^{n} \lambda_j \boldsymbol{U}_{:,k} (\boldsymbol{U}_{:,k})^T \boldsymbol{X} \boldsymbol{V} \\ &= \sum_{k=1}^{n} \boldsymbol{U}_{:,k} (\boldsymbol{U}_{:,k})^T \boldsymbol{X} \boldsymbol{W}^{(k)} \\ &= \boldsymbol{\Theta}_{\text{GCN}} * \boldsymbol{X} \end{aligned} \tag{10}$$

where $\boldsymbol{W}^{(k)} = \lambda_j \boldsymbol{V}$ and corresponding $\boldsymbol{\Theta}_{\text{GCN}} \in \mathbb{R}^{n \times c \times d}$.

As a first step, the MIMO-GC helps us with the understanding of properties of various approximations and can consequently improve these approximations. Based on Example 1, the GCN utilizes a single shared parameter matrix $\boldsymbol{V}$ across all components. Each component is then amplified according to its respective eigenvalue, which is shared across all combinations of input and output channels. Other message-passing operations may utilize a different matrix $\tilde{\boldsymbol{A}}$ instead of $\boldsymbol{A}_{\text{sym}}$. However, as using any single computational graph $\tilde{\boldsymbol{A}}$ can be similarly decomposed, the amplification of components is fixed and shared across all feature channels for any given $\tilde{\boldsymbol{A}}$. We refer to this phenomenon as shared component amplification (SCA) When repeatedly applying such filters or message-passing operations, SCA leads to the well-known phenomenon of over-smoothing and, more generally, rank collapse [23,30]. We provide further details on this phenomenon in our appendix.

Contrarily, the MIMO-GC requires multiple computational graphs to amplify different components across feature channels. Equivalently, applying distinct feature transformations for each node pair can improve approximations. Developing approximations with these properties can lead to more effective learning on graph-structured data.

4 Localized MIMO Graph Convolutions

Based on Eq. (7b), we localize the MIMO-GC by aggregating over the neighboring nodes instead of all nodes of a given graph:

Definition 1. *We define the Localized MIMO Graph Convolution (LMGC) as:*

$$\boldsymbol{x}'_{(i)} = \sum_{v_j \in N_i} \boldsymbol{W}_{(i,j)} \boldsymbol{x}_{(j)} \tag{11a}$$

$$= \left[\sum_{k \in [K]} \tilde{\boldsymbol{A}}^{(k)} \boldsymbol{X} \boldsymbol{W}^{(k)} \right]_{i,:} \tag{11b}$$

where $K \in \mathbb{N}$ and each $\boldsymbol{W}_{(i,j)} = \sum_{k \in [K]} \alpha_{(k)}^{(i,j)} \boldsymbol{W}^{(k)} \in \mathbb{R}^{c \times d}$ is linear combination based on $\alpha_{(1)}^{(i,j)}, \dots \alpha_{(K)}^{(i,j)} \in \mathbb{R}$ and $\boldsymbol{W}^{(1)}, \dots, \boldsymbol{W}^{(K)} \in \mathbb{R}^{d \times c}$. The entries $A_{i,j}^{(k)} = \alpha_{(k)}^{(i,j)}$ are given by the corresponding coefficients.

In this definition, the number of terms K and the coefficients or edge weights $\alpha_{(k)}^{(i,j)}$ can be freely chosen, which allows methods that do not use the expensive eigenvector computation. The LMGC is permutation equivariant if the coefficients $\alpha_{(k)}^{(i,j)}$ are also equivariant, for example, when derived from a function of the nodes v_i and v_j. The LMGC can also be applied across different graphs and for directed graphs. As with MIMO-GCs, the LMGC can be equivalently restated as operating on K computational graphs. The edge weights of the k-th computational graph are given by $\alpha_{(k)}^{(i,j)}$. Consequently, the LMGC can represent many linear MPNNs for different values for $\alpha_{(k)}^{(i,j)}$. We provide three examples below:

Example 2 (GCN [16]). Let $\boldsymbol{V} \in \mathbb{R}^{c \times d}$ be a feature transformation. The update step

$$\boldsymbol{x}'_{(i)} = \sum_{v_j \in N_i} \frac{1}{\sqrt{d_i}\sqrt{d_j}} \boldsymbol{V} \boldsymbol{x}_{(j)} \tag{12}$$

is an LMGC with $K = 1$, $\boldsymbol{W}^{(1)} = \boldsymbol{V}$, and $\alpha_{(1)}^{(i,j)} = \frac{1}{\sqrt{d_i}\sqrt{d_j}}$ where $d_i, d_j \in \mathbb{N}$ are the degrees of nodes v_i and v_j, respectively.

As the MIMO-GC is similar to multi-head self-attention, the LMGC is related to local multi-head attention-based methods while allowing for more flexible attention scores, i.e., scores do not need to sum to one for every node:

Example 3 (GAT [37]). Let H be the number of heads, $\boldsymbol{V}^{(h)}$ the linear transformation of head $h \in [H]$, and $a_{(h)}^{(i,j)} \in \mathbb{R}$ the attention score between nodes v_i and v_j. The update step

$$\boldsymbol{x}'_{(i)} = \sum_{h \in [H]} \sum_{v_j \in N_i} a_{(h)}^{(i,j)} \boldsymbol{V}^{(h)} \boldsymbol{x}_{(j)} \tag{13}$$

is an LMGC with $K = H$, $\boldsymbol{W}^{(h)} = \boldsymbol{V}^{(h)}$ and $\alpha_{(h)}^{(i,j)} = a_{(h)}^{(i,j)}$.

The LMGC can also represent gating mechanisms, e.g., the GatedGCN [7] or neural sheaf diffusion [12].

The general form of the LMGC allows for a more focused development of novel and powerful methods. With specific choices of $\alpha_{(k)}^{(i,j)}$, the LMGC can model a symmetric or directed flow of information and can construct anisotropic or isotropic messages.

Theoretical Properties. Studying theoretical properties of LMGCs reduces to studying the effects of coefficients $\alpha_{(k)}^{(i,j)}$. For example, the LMGC cannot represent non-linear feature transformations, which are typically used to ensure injectivity, e.g., by GIN [39]. This allows GNNs to match the expressivity of the Weisfeiler-Leman graph isomorphism test [19], a key property for graph-level tasks. However, we find that any LMGC with $K > 0$ computational graphs is also injective for almost every choice coefficients $\alpha_{(k)}^{(i,j)}$ without requiring a non-linear feature transformation:

Proposition 3 (Injectivity). *Let $f(\boldsymbol{x}_{(i)}, N_i) = \sum_{\boldsymbol{x}_{(j)} \in N_i} \boldsymbol{W}_{(i,j)} \boldsymbol{x}_{(j)}$ be an LMGC with $K \geq 1$ and $\mathcal{X}$ a countable set. Then, $f(\boldsymbol{x}_{(p)}, \mathcal{X}_p)$ is injective for finite multisets $\mathcal{X}_p \subset \mathcal{X}$ and elements $\boldsymbol{x}_{(p)} \in \mathcal{X}$ for a.e. choice of coefficients $\alpha_{(k)}^{(i,j)}$ and a.e. $\boldsymbol{W}^{(k)}$ for all $k \in [K]$.*

Different components can be amplified across feature channels when further using $K > 1$ computational graphs. The resulting node representations are linearly independent for almost every choice of coefficients $\alpha_{(k)}^{(i,j)}$. This prevents the shared component amplification of methods utilizing a single computational graph.

Proposition 4 (Linear Independence). *Let $f(\boldsymbol{x}_{(i)}, N_i) = \sum_{\boldsymbol{x}_{(j)} \in N_i} \boldsymbol{W}_{(i,j)} \boldsymbol{x}_{(j)}$ be an LMGC with $K > 1$ and $\mathcal{X}$ a countable set. Then, $f(\boldsymbol{x}_{(i)}, \mathcal{X}_1)$ is linearly independent to $f(\boldsymbol{x}_{(j)}, \mathcal{X}_2)$ for all finite multisets $\mathcal{X}_1, \mathcal{X}_2 \subset \mathcal{X}$ with $\mathcal{X}_1 \neq c \cdot \mathcal{X}_2$ for any $c \in \mathbb{N}$ and elements $\boldsymbol{x}_{(i)}, \boldsymbol{x}_{(j)} \in \mathcal{X}$ for a.e. choice of coefficients $\alpha_{(k)}^{(i,j)}$ and a.e. $\boldsymbol{W}^{(k)}$ for all $k \in [K]$.*

This result aligns with previous findings that identified cases where multiple computational graphs can ensure linearly independent representations [28]. Importantly, each $\alpha_{(k)}^{(i,j)}$ can be independently obtained, e.g., by a function $\alpha_{(k)}^{(i,j)} = \phi_k(\boldsymbol{x}_{(i)}, \boldsymbol{x}_{(j)}) \in \mathbb{R}$ of the corresponding node states. Many functions ϕ_k satisfy Proposition 3 and Proposition 4. A neural network can then approximate such a function. As a negative example of such a functions, softmax-activated attention scores do not satisfy the a.e. condition as the space of scores forms a measure-zero set, e.g., for GAT [37] and the more powerful GATv2 [3]. As has been pointed out by several works [39], such methods cannot distinguish multisets of different multiplicities, e.g., when $\mathcal{X}_1 = \{\{\boldsymbol{x}_1\}\}$ and $\mathcal{X}_2 = \{\{\boldsymbol{x}_1, \boldsymbol{x}_1\}\}$. Other methods, such as FAGCN [1] and GGCN [41], proposed to apply the tanh activation function instead, which does not constrain the outputs to a measure-zero set.

Thus, an LMGC can incorporate the advantages of attention-based by filtering incoming messages and preventing the shared component amplification across feature channels by utilizing multiple computational graphs. At the same time, it applies linear feature transformations and can be injective on multisets, as in GIN.

An LMGC Instantiation. When constructing an LMGC instantiation, only the number of computational graphs K and the coefficients $\alpha_{(k)}^{(i,j)}$ for all $k \in [K]$ need to be defined. For our empirical study, we define a simple LMGC instantiation as a mix of GATv2 and

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 1: Test MAE results on ZINC12k. LapPE indicates that a Laplacian position encoding is concatenated to the initial features. For Jumping Knowledge, the channel-wise maximum value after each iteration is used for each after the message-passing steps. Residual indicates that the input to each message-passing step is added to its output. With + All three, these three techniques are simultaneously applied. Best scores in **bold**, second-best underlined.

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 2: Test accuracy on heterophilic node classification tasks. Best scores in **bold**, second-best underlined. All models contain at most 100 000 parameters and the same hyperparameter optimization was applied.

FAGCN. We define the coefficients as

$$\alpha_{(k)}^{(i,j)} = \phi_k(\boldsymbol{x}_{(i)}, \boldsymbol{x}_{(j)}) \coloneqq \sigma_2(\boldsymbol{v}_{(k)}^T \sigma_1(\boldsymbol{W}^{(1)}\boldsymbol{x}_{(i)} || \ldots || \boldsymbol{W}^{(K)}\boldsymbol{x}_{(i)} || \boldsymbol{W}^{(1)}\boldsymbol{x}_{(j)} || \ldots || \boldsymbol{W}^{(K)}\boldsymbol{x}_{(j)})) \quad (14)$$

where $\boldsymbol{v}_{(k)} \in \mathbb{R}^{2 \cdot K \cdot c}$ are learnable vectors for $k \in [K]$, σ_1 is the LeakyReLU activation and σ_2 is the tanh activation function. The execution time is slightly favorable compared to GATv2, as we do not normalize the messages.

5 Related Work

We now describe previous works related to various parts of the MIMO-GC and the LMGC.

Graph Convolutions. Bruna et al. [4] extend the SISO-GC to the MIMO case by utilizing a filter between all pairs of input and output channels. This extension is equivalent to the MIMO-GC directly derived through the convolution theorem. Approximations are derived in the SISO case and mapped to the MIMO case using the same procedure afterward. Hammond et al. [11] propose to approximate the SISO-GC using Chebyshev polynomials in the SISO case. Defferrard et al. [6] employ separate filters for pairs

of input and output channels to extend Chebyshev polynomials to the MIMO case. Sandryhaila [32] define general polynomial graph filters for the SISO case. Using the same procedure, Gama et al. [10] extend these polynomial graph filters to the MIMO case. Kipf and Welling [16] derive the GCN as a 1-localized approximation of the SISO Chebyshev polynomials. They equivalently extend it to the MIMO case afterward by applying separate parameters for each combination of input and output channels. Most other MPNNs are derived from the GCN to mitigate various shortcomings [37,39,29]. Directly approximating the MIMO-GC allows us to benefit from MIMO-specific properties of the graph convolution.

MIMO Improvements. While most MPNNs are applied to the MIMO case, many of these are well-known to be unable to amplify distinct components across channels, a phenomenon known as over-smoothing [23], over-correlation [15], or rank collapse [30,27]. Various methods have been proposed to improve multi-channel learning within MPNNs. Luan et al. [22] propose to apply separate graph filters for different feature channels. In ADR-GNNs [9], feature channels are separately aggregated using channel-specific edge weights. Other works similarly propose to apply distinct filters across channels [21]. Zhou et al. [42] propose the multi-channel graph neural network that obtains multiple computational graphs through a pooling operation and learns interaction scores between graphs. Utilizing multiple computational graphs has been extensively studied in mitigating over-smoothing and representational rank collapse [28]. Applying different linear transformations between pairs of nodes has also been derived within neural sheaf diffusion [12,2]. As the MIMO-GC and LMGC naturally allow multi-channel learning, these frameworks can be closer aligned as approximations of the MIMO-GC. The LMGC can equivalently be interpreted as message-passing on multigraphs. Butler et al. [5] introduced convolutional multigraph neural networks that utilize polynomial filters on multigraphs.

Approaches Related to the LMGC. The LMGC is closely related to several existing methods. As described in Example 3, multi-head attention-based methods like GAT [37] and GATv2 [3] are LMGCs with constraints on the attention scores by applying the softmax activation. By lifting this constraint, LMGCs can be injective on multisets (Proposition 3). Several other methods have been proposed to replace the softmax activation. The FAGCN [1] instead applies the tanh activation function to amplify high-frequencies or low-frequencies. Similarly, the GGCN [41] allows learning of signed edge weights. Other studies considered replacing the softmax activation function within transformers and self-attention modules. Wortsman et al. [38] apply the ReLU activation in vision transformers. Saratchandran et al. [33] found empirical success using polynomial activation functions for self-attention. However, as self-attention typically considers a fully connected graph, these works did not study distinguishing structural differences. Contrarily, Proposition 3 shows that differences in the number of neighbors can be distinguished without the softmax activation.

6 Experiments

We now want to confirm the beneficial properties of LMGCs. As the LMGC can match the expressive power of GIN, we want to evaluate whether it can match the performance of GIN for graph-level tasks. We also evaluate whether the LMGC can match the performance of attention-based methods for node-level tasks. All experiments are run on an H100 GPU. Additional details on all models, datasets, and hyperparameters are provided as supplementary material.[1]

6.1 Methods

We consider the following four message-passing methods across all experiments. We conduct all results ourselves using the same hyperparameter ranges across methods.

GATv2. This method [3] corresponds to an LMGC with $\alpha_{(k)}^{(i,j)} = \sigma_2(\boldsymbol{v}_{(k)}^T \sigma_1(\boldsymbol{W}^{(k)}\boldsymbol{x}_{(i)} + \boldsymbol{W}^{(k)}\boldsymbol{x}_{(j)}))$ where σ_1 is the LeakyReLU activation function and σ_2 is the node-wise softmax activation function and $\boldsymbol{v}^{(k)} \in \mathbb{R}^c$ is a learnable vector. We set the number of heads to $K = 4$ for all experiments.

FAGCN. This method was designed for heterophilic node classification tasks by allowing for negative edge weights [1]. Stated in the LMGC framework, we evaluate a method that sets $K = 1$ and $\alpha_{(1)}^{(i,j)} = \frac{\sigma(\boldsymbol{v}[\boldsymbol{x}_{(i)} || \boldsymbol{x}_{(j)}])}{\sqrt{d_i}\sqrt{d_j}}$ where σ is the tanh activation function, $\boldsymbol{v} \in \mathbb{R}^{2 \cdot d}$ is a learnable vector and d_i, d_j are the degrees of nodes v_i and v_j, respectively. FAGCN is not always injective due to the degree normalization.

ACM. Written in the LMGC framework, the adaptive channel mixing (ACM) [22] proposes to utilize $\tilde{\boldsymbol{A}}^{(1)} = \boldsymbol{A}_{\text{sym}}$ for amplifying low-frequency components and $\tilde{\boldsymbol{A}}^{(2)} = \boldsymbol{L}_{\text{sym}}$ for amplifying high-frequency components. They further propose a third computational graph $\tilde{\boldsymbol{A}}^{(3)} = \boldsymbol{I}$, which we utilize whenever residual connections are used.

GIN. For graph-level tasks, the GIN [39] is particularly effective as it can match the expressivity of the WL-test due to the non-linear feature transformation. As the non-linear feature transformation, we apply a two-layer MLP with ReLU activations.

LMGC. As the LMGC can combine the favorable properties of the other three methods, we utilize the instantiation of the LMGC as described in Eq. (14). As with GATv2, we set the total number of heads to $K = 4$ for all experiments.

6.2 Graph-Level Prediction

GIN is typically used for graph-level tasks due to its expressive power. Based on Proposition 3, we now want to validate that the LMGC can match these results empirically.

[1] Our implementation is available at https://github.com/roth-andreas/mimo-graph-convolutions.

We consider the challenging ZINC12k dataset [34]. It consists of around $12\,000$ molecular graphs, with the task being to predict the constrained solubility of each molecule. We integrate all models into the implementation of GraphGPS [26] and the Long Range Graph Benchmark [8]. Based on Toenshoff et al. [35], we optimize the number of layers in $\{6, 8, 10\}$ and the learning rate in $\{0.001, 0.0003, 0.0001\}$ using a grid search. Each model utilizes at most $100\,000$ parameters to ensure fairness.

In Table 1, we present results of a detailed study in which we combine these base message-passing methods with various other established techniques. These techniques are Laplacian positional encoding (LapPE) [18], jumping knowledge [40] and residual connections [13]. We find all methods to benefit from these techniques, with the LMGC achieving the best results in all cases. We provide additional results, including runtimes and training losses, as supplementary material.

6.3 Node Classification

While expressivity is a key property for graph-level tasks, attention-based methods typically outperform GIN on node-level tasks due to their ability to filter messages [3]. Thus, we also evaluate whether the LMGC can match the performance of GATv2 and FAGCN on these tasks. We consider six heterophilic benchmark datasets for node classification: Texas, Cornell, Wisconsin, Film, Chameleon, and Squirrel. We use the ten splits into train, validation, and test sets proposed by Pei et al. [25]. We integrate all models into the implementation from Rusch et al. [31]. As with ZINC, each model uses at most $100\,000$ parameters. For each method, we tune the learning rate in $\{0.01, 0.003, 0.001\}$ and dropout ratio in $\{0.0, 0.25, 0.5\}$ using a grid search, as these affected the results the most. Based on the optimal hyperparameters for the validation set, we rerun each method five times for all ten splits and report average test results.

These average test accuracies are presented in Table 2. GIN achieves the lowest accuracy, and LMGC achieves the highest accuracy across all tasks. While the differences are only a few percentage points, these experiments confirm that the LMGC can combine the benefits of GATv2, FAGCN, GIN, and the MIMO-GC into a single model.

7 Conclusion

This work derives the MIMO graph convolution (MIMO-GC) using the convolution theorem and emphasizes the advantages of approximating the graph convolution in the MIMO case rather than the SISO case. A key property of the MIMO-GC is operating on multiple computational graphs or equivalently applying distinct linear transformations for each node pair. We have proven that the localized form is injective and results in linearly independent representations for almost every choice of edge weights. Due to our direct theoretical derivation from the MIMO-GC and the generality of the LMGC framework, studying properties of message-passing operations can now focus on analyzing the coefficients $\alpha_{(k)}^{(i,j)}$. This allows the development of more effective methods within a well-defined framework. While we have confirmed the advantages and potential of the LMGC framework, identifying optimal instantiations of LMGCs for specific tasks remains open.

Acknowledgments. Part of this research has been funded by the Federal Ministry of Education and Research of Germany and the state of North-Rhine Westphalia as part of the Lamarr-Institute for Machine Learning and Artificial Intelligence and by the Federal Ministry of Education and Research of Germany under grant no. 01IS22094E WEST-AI. Simulations were performed with computing resources granted by WestAI under project rwth1631.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

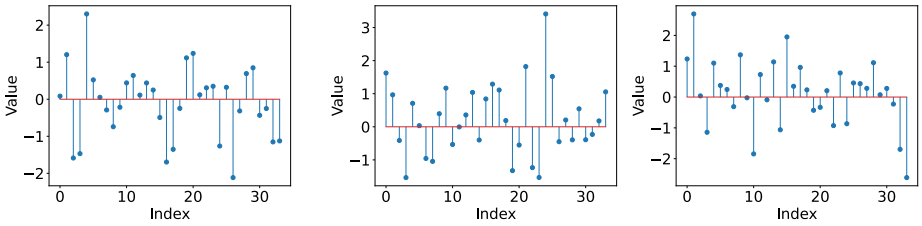
1. Bo, D., Wang, X., Shi, C., Shen, H.: Beyond low-frequency information in graph convolutional networks. In: Thirty-Fifth AAAI Conference on Artificial Intelligence, Virtual Event, February 2-9. pp. 3950–3957. AAAI Press (2021). https://doi.org/10.1609/AAAI.V35I5.16514
2. Bodnar, C., Giovanni, F.D., Chamberlain, B.P., Lió, P., Bronstein, M.M.: 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, New Orleans, LA, USA, November 28 - December 9 (2022)
3. Brody, S., Alon, U., Yahav, E.: How attentive are graph attention networks? In: The Tenth International Conference on Learning Representations, Virtual Event, April 25-29 (2022)
4. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. In: 2nd International Conference on Learning Representations, Banff, AB, Canada, April 14-16, Conference Track Proceedings (2014)
5. Butler, L., Parada-Mayorga, A., Ribeiro, A.: Convolutional learning on multigraphs. IEEE Trans. Signal Process. **71**, 933–946 (2023). https://doi.org/10.1109/TSP.2023.3259144
6. Defferrard, M., Bresson, X., Vandergheynst, P.: 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, Barcelona, Spain. pp. 3837–3845 (2016)
7. Dwivedi, V.P., Joshi, C.K., Luu, A.T., Laurent, T., Bengio, Y., Bresson, X.: Benchmarking graph neural networks. J. Mach. Learn. Res. **24**, 43:1–43:48 (2023)
8. Dwivedi, V.P., Rampásek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A.T., Beaini, D.: Long range graph benchmark. In: Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, New Orleans, LA, USA, November 28 - December 9 (2022)
9. Eliasof, M., Haber, E., Treister, E.: ADR-GNN: advection-diffusion-reaction graph neural networks. CoRR **abs/2307.16092** (2023). https://doi.org/10.48550/ARXIV.2307.16092
10. Gama, F., Marques, A.G., Ribeiro, A., Leus, G.: MIMO graph filters for convolutional neural networks. In: 19th IEEE International Workshop on Signal Processing Advances in Wireless Communications, Kalamata, Greece, June 25-28. pp. 1–5. IEEE (2018). https://doi.org/10.1109/SPAWC.2018.8445934
11. Hammond, D.K., Vandergheynst, P., Gribonval, R.: Wavelets on graphs via spectral graph theory. Applied and Computational Harmonic Analysis **30**(2), 129–150 (2011)
12. Hansen, J., Gebhart, T.: Sheaf neural networks. CoRR **abs/2012.06333** (2020)

13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, June 27-30. pp. 770–778. IEEE Computer Society (2016). https://doi.org/10.1109/CVPR.2016.90
14. Hu, W., Fey, M., Ren, H., Nakata, M., Dong, Y., Leskovec, J.: 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, December 2021, virtual (2021)
15. Jin, W., Liu, X., Ma, Y., Aggarwal, C.C., Tang, J.: 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. pp. 709–719. ACM (2022). https://doi.org/10.1145/3534678.3539445
16. Kipf, T.N., Welling, M.: 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 (2017)
17. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009). https://doi.org/10.1137/07070111X
18. Kreuzer, D., Beaini, D., Hamilton, W.L., Létourneau, V., Tossou, P.: Rethinking graph transformers with spectral attention. In: Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, December 6-14, virtual. pp. 21618–21629 (2021)
19. Leman, A., Weisfeiler, B.: A reduction of a graph to a canonical form and an algebra arising during this reduction. Nauchno-Technicheskaya Informatsiya **2**(9), 12–16 (1968)
20. Levie, R., Monti, F., Bresson, X., Bronstein, M.M.: Cayleynets: Graph convolutional neural networks with complex rational spectral filters. IEEE Trans. Signal Process. **67**(1), 97–109 (2019). https://doi.org/10.1109/TSP.2018.2879624
21. Liu, X., Ng, A.H., Lei, F., Zhang, Y., Li, Z.: Gpnet: Simplifying graph neural networks via multi-channel geometric polynomials. Inf. Sci. **694**, 121696 (2025)
22. Luan, S., Hua, C., Lu, Q., Zhu, J., Zhao, M., Zhang, S., Chang, X., Precup, D.: Revisiting heterophily for graph neural networks. In: Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, New Orleans, LA, USA, November 28 - December 9 (2022)
23. Oono, K., Suzuki, T.: Graph neural networks exponentially lose expressive power for node classification. In: 8th International Conference on Learning Representations Addis Ababa, Ethiopia, April 26-30 (2020)
24. O'Neil, R.: Convolution operators and $L(p, q)$ spaces. Duke Mathematical Journal **30**(1), 129 – 142 (1963). https://doi.org/10.1215/S0012-7094-63-03015-1
25. Pei, H., Wei, B., Chang, K.C., Lei, Y., Yang, B.: Geom-gcn: Geometric graph convolutional networks. In: 8th International Conference on Learning Representations, Addis Ababa, Ethiopia, April 26-30 (2020)
26. Rampásek, L., Galkin, M., Dwivedi, V.P., Luu, A.T., Wolf, G., Beaini, D.: Recipe for a general, powerful, scalable graph transformer. In: Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, New Orleans, LA, USA, November 28 - December 9 (2022)
27. Roth, A.: Simplifying the theory on over-smoothing. CoRR **abs/2407.11876** (2024). https://doi.org/10.48550/ARXIV.2407.11876
28. Roth, A., Bause, F., Kriege, N.M., Liebig, T.: Preventing representational rank collapse in mpnns by splitting the computational graph. CoRR **abs/2409.11504** (2024)
29. Roth, A., Liebig, T.: 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. vol. 13714, pp.

469–484. Springer (2022). https://doi.org/10.1007/978-3-031-26390-3_27
30. Roth, A., Liebig, T.: Rank collapse causes over-smoothing and over-correlation in graph neural networks. In: Learning on Graphs Conference, 27-30 November 2023, Virtual Event. vol. 231, p. 35. PMLR (2023)
31. Rusch, T.K., Chamberlain, B.P., Mahoney, M.W., Bronstein, M.M., Mishra, S.: Gradient gating for deep multi-rate learning on graphs. In: The Eleventh International Conference on Learning Representations, Kigali, Rwanda, May 1-5 (2023)
32. Sandryhaila, A., Moura, J.M.: Discrete signal processing on graphs. IEEE transactions on signal processing **61**(7), 1644–1656 (2013)
33. Saratchandran, H., Zheng, J., Ji, Y., Zhang, W., Lucey, S.: Rethinking softmax: Self-attention with polynomial activations. CoRR **abs/2410.18613** (2024). https://doi.org/10.48550/ARXIV.2410.18613
34. Sterling, T., Irwin, J.J.: ZINC 15 - ligand discovery for everyone. J. Chem. Inf. Model. **55**(11), 2324–2337 (2015). https://doi.org/10.1021/ACS.JCIM.5B00559
35. Tönshoff, J., Ritzert, M., Rosenbluth, E., Grohe, M.: Where did the gap go? reassessing the long-range graph benchmark. Trans. Mach. Learn. Res. **2024** (2024)
36. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, Long Beach, CA, USA. pp. 5998–6008 (2017)
37. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: 6th International Conference on Learning Representations, Vancouver, BC, Canada, April 30 - May 3 (2018)
38. Wortsman, M., Lee, J., Gilmer, J., Kornblith, S.: Replacing softmax with relu in vision transformers. CoRR **abs/2309.08586** (2023). https://doi.org/10.48550/ARXIV.2309.08586
39. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: 7th International Conference on Learning Representations, New Orleans, LA, USA, May 6-9 (2019)
40. Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. In: Proceedings of the 35th International Conference on Machine Learning, Stockholmsmässan, Stockholm, Sweden, July 10-15. vol. 80, pp. 5449–5458. PMLR (2018)
41. Yan, Y., Hashemi, M., Swersky, K., Yang, Y., Koutra, D.: Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In: IEEE International Conference on Data Mining, Orlando, FL, USA, November 28 - Dec. 1. pp. 1287–1292. IEEE (2022). https://doi.org/10.1109/ICDM54844.2022.00169
42. Zhou, K., Song, Q., Huang, X., Zha, D., Zou, N., Hu, X.: Multi-channel graph neural networks. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. pp. 1352–1358. ijcai.org (2020). https://doi.org/10.24963/IJCAI.2020/188

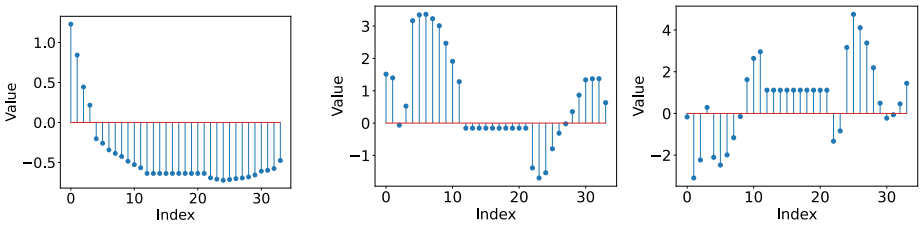
A Further Details on Shared Component Amplification (SCA)

We provide further details on shared component amplification, which we used throughout the main paper. Based on the graph Fourier transformed graph convolution $F(\boldsymbol{\theta} *$



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

Fig. 2: Examples of random filters $F(\boldsymbol{\theta})$ in the graph Fourier domain.

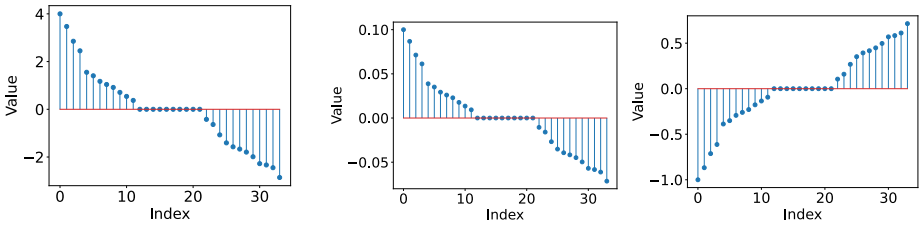


(a) $K = 2$. (b) $K = 8$. (c) $K = 16$.

Fig. 3: Spectral filters based on Chebyshev polynomials $F(\boldsymbol{\theta}) = \sum_{k=0}^{K} w_k T_k(\boldsymbol{\lambda})$ of different degrees K.

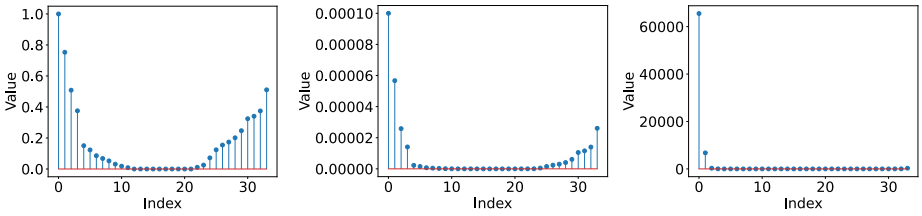
$\boldsymbol{x}) = F(\boldsymbol{\theta}) \odot F(\boldsymbol{x})$, we visualize the spectral filter $F(\theta)$ for the general graph convolution, polynomial approximations, and the GCN as a first-order approximation. In Figure 2, we visualize $F(\theta)$ for random filters θ, as is allowed for the general graph convolution (Equation 1). Polynomial filters define $F(\theta)$ as a polynomial function of the eigenvalues of the normalized adjacency matrix or the corresponding graph Laplacian (Equation 2). We visualize random such filters based on Chebyshev polynomials of different degrees in Figure 3 and visually observe this polynomial structure.

The GCN as a first-order polynomial (Equation 3) can similarly be described as a spectral filter $F(\theta) = w\boldsymbol{\lambda}$ where $w \in \mathbb{R}$ is the sole parameter. We visualize this filter



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

Fig. 4: Filters of the GCN for $F(\boldsymbol{\theta}) = w\boldsymbol{\lambda}$ with different values for w.



(a) $k = 2$. (b) $k = 4$. (c) $k = 16$.

Fig. 5: Combination of k repetitions of random GCN filters $F(\boldsymbol{\theta}) = w_k\boldsymbol{\lambda} \odot \cdots \odot w_1\boldsymbol{\lambda}$ with random values for $w_1, \ldots, w_k$.

Task	Graph Regression		Node Classification					
Dataset	ZINC	ZINC12k	Texas	Cornell	Wisconsin	Film	Chameleon	Squirrel
# of graphs	249 456	12 000	1	1	1	1	1	1
avg. # of nodes	~ 23.2	~ 23.2	183	183	251	7600	2277	5201
avg. # of edges	~ 49.8	~ 49.8	325	298	515	30 019	36 101	217 073
avg. node degree	~ 2.1	~ 2.1	~ 1.8	~ 1.6	~ 2.1	~ 3.9	~ 15.9	~ 41.7
# of node features	1	1	1703	1703	1703	932	2325	2089
# of classes	1	1	5	5	5	5	5	5

Table 3: Statistics of all utilized datasets.

for different values w in Figure 4. As all filters $w\boldsymbol{\lambda}$ are equivalent up to scaling, we also observe the visual equivalence of such filters. For any parameter w, each component gets amplified in the same way. For any other fixed $\boldsymbol{\lambda}$, any parameter w would still lead to the same amplification of components. In the MIMO case, such a filter $w_{(i,j)}\boldsymbol{\lambda}$ is applied for each combination of input channel i and output channel j. As for any $w_{(i,j)}$, the components are amplified in the same way based on the given $\boldsymbol{\lambda}$, we refer to this phenomenon as shared component amplification.

When such a filter is applied repeatedly for random GCN filters, the component corresponding to the maximal absolute value in $\boldsymbol{\lambda}$ dominates all other components exponentially. Repeated applications of GCN filters $F(\boldsymbol{\theta}_K)\odot\cdots\odot F(\boldsymbol{\theta}_1)$ are visualized in Figure 5. We refer to this as component dominance. In the MIMO case, when we have

Method	Plain	+ LapPE	+ Jumping Knowledge	+ Residual	+ All three
GATv2	8/0.001	6/0.001	10/0.001	10/0.001	10/0.001
FAGCN	8/0.001	8/0.001	10/0.001	10/0.001	10/0.001
ACM	8/0.001	8/0.001	10/0.001	10/0.001	10/0.001
GIN	8/0.0003	6/0.001	8/0.0003	10/0.0003	10/0.001
LMGC	6/0.001	8/0.001	8/0.001	10/0.001	10/0.001

Table 4: Optimal hyperparameters for the experiments presented in Table 1. Each entry describes the optimal 'number of layers / learning rate'.

Method	Texas	Cornell	Wisconsin	Film	Chameleon	Squirrel
GATv2	0.01/0.25	0.01/0.25	0.01/0.25	0.01/0.5	0.01/0.25	0.01/0.0
FAGCN	0.003/0.5	0.01/0.25	0.01/0.25	0.001/0.5	0.001/0.25	0.01/0.25
ACM	0.01/0.25	0.01/0.25	0.003/0.25	0.01/0.5	0.01/0.25	0.01/0.25
GIN	0.01/0.25	0.01/0.25	0.01/0.25	0.01/0.5	0.003/0.25	0.01/0.25
LMCGC	0.01/0.25	0.003/0.25	0.003/0.25	0.001/0.5	0.003/0.0	0.001/0.5

Table 5: Optimal hyperparameters for the experiments presented in Table 2. Each entry describes the optimal 'learning rate / dropout rate'.

	ZINC			ZINC12k		
Method	Train	Test	Time per Epoch (s)	Train	Test	Time per Epoch (s)
GATv2	0.077 ± 0.001	0.114 ± 0.004	52.7 ± 0.0	$\underline{0.005} \pm 0.003$	0.377 ± 0.024	2.5 ± 0.1
FAGCN	0.093 ± 0.008	0.130 ± 0.003	52.9 ± 1.1	0.016 ± 0.006	0.365 ± 0.018	2.5 ± 0.0
ACM	0.109 ± 0.003	0.128 ± 0.001	73.1 ± 0.9	0.019 ± 0.005	0.278 ± 0.006	3.4 ± 0.0
GIN	$\underline{0.068} \pm 0.001$	$\underline{0.088} \pm 0.002$	$\mathbf{35.5} \pm 1.0$	0.018 ± 0.009	$\underline{0.272} \pm 0.009$	$\mathbf{1.7} \pm 0.0$
LMGC	$\mathbf{0.054} \pm 0.001$	$\mathbf{0.080} \pm 0.001$	$\underline{47.5} \pm 0.5$	$\mathbf{0.002} \pm 0.001$	$\mathbf{0.241} \pm 0.018$	$\underline{2.4} \pm 0.3$

Table 6: MAE results on ZINC and ZINC12k. Optimal hyperparameters for train and test results are independently obtained. Best scores in **bold**, second-best underlined. Each model uses at most 100 000 parameters.

shared component amplification, the same component also dominates each combination of input and output channels with increasingly many repetitions. This combination is equivalent to rank collapse and over-smoothing when a smooth component is amplified.

B Mathematical Details

In this section, we provide the proofs for all statements in the main paper.

B.1 Proofs for Section 3.

Proof of Theorem 1.

Proof. We use the vectorized signal $\hat{\boldsymbol{x}} = \mathrm{vec}(\boldsymbol{X}) \in \mathbb{R}^{n \cdot c}$ by stacking its columns. The graph Fourier transform on matrices and tensors is applied along the node dimension, i.e., independently on each channel. For matrix $\boldsymbol{X}$ this results in

$$F(\boldsymbol{X}) = \boldsymbol{U}^T \boldsymbol{X} \in \mathbb{R}^{n \times d}$$

and for tensor $\mathbf{W}$ in

$$\hat{\mathbf{W}} = F(\mathbf{W}) = \boldsymbol{U}^T \times_1 \mathbf{W} \in \mathbb{R}^{n \times c \times d}$$

where $\times_1$ is the 1-mode tensor matrix product [17] that performs the desired broadcasted matrix multiplication. With this, we state the multi-channel graph convolution in the Fourier domain as

	ZINC		ZINC	
Method	Train	Test	Train	Test
GATv2	10/0.001	10/0.001	8/0.001	8/0.001
FAGCN	8/0.001	8/0.001	6/0.001	8/0.001
ACM	6/0.001	6/0.001	8/0.001	8/0.001
GIN	6/0.0003	6/0.0003	6/0.001	8/0.0003
LMGC	8/0.001	8/0.001	6/0.001	6/0.001

Table 7: Optimal hyperparameters for the experiments presented in Table 6. Each entry describes the optimal 'number of layers / learning rate'.

$$\mathbf{W} * \boldsymbol{X} = \boldsymbol{U}(\boldsymbol{U}^T \times_1 \mathbf{W} \odot \boldsymbol{U}^T \boldsymbol{X}) = \boldsymbol{U}(\hat{\mathbf{W}} \odot \boldsymbol{U}^T \boldsymbol{X})\,.$$

The element-wise product $\hat{\mathbf{W}}_k(\boldsymbol{U}^T\boldsymbol{X})_k$ is a matrix-vector product. Similarly to the SISO-GC, we simplify this expression using matrix multiplications. Equivalently to the SISO-GC, this can be achieved by diagonalizing $\hat{\mathbf{W}}$ into a block matrix of diagonal blocks

$$\boldsymbol{D} = \begin{bmatrix} \hat{W}_{1,0,0} & 0 & 0 & & \hat{W}_{1,0,d} & 0 & 0 \\ 0 & \ddots & 0 & \dots & 0 & \ddots & 0 \\ 0 & 0 & \hat{W}_{n,0,0} & & 0 & 0 & \hat{W}_{n,0,d} \\ & \vdots & & \ddots & & \vdots & \\ \hat{W}_{1,c,0} & 0 & 0 & & \hat{W}_{1,c,d} & 0 & 0 \\ 0 & \ddots & 0 & \dots & 0 & \ddots & 0 \\ 0 & 0 & \hat{W}_{n,c,0} & & 0 & 0 & \hat{W}_{n,c,d} \end{bmatrix}$$

$\in \mathbb{R}^{nc \times nd}$ where $\hat{W}_{k,i,j} \in \mathbb{R}$. This simplifies the equivalent vectorized form into

$$\text{vec}(\hat{\mathbf{W}} \odot \boldsymbol{U}^T\boldsymbol{X}) = \boldsymbol{D}\text{vec}(\boldsymbol{U}^T\boldsymbol{X}) = \boldsymbol{D}\left(\boldsymbol{I}_d \otimes \boldsymbol{U}^T\right)\text{vec}(\boldsymbol{X})$$

by utilizing the Kronecker product $\otimes$. The matrix $\boldsymbol{D}$ can further be decomposed into a sum of Kronecker products $\boldsymbol{D} = \sum_{k=1}^{n} \hat{\mathbf{W}}_i \otimes \boldsymbol{I}_n^{(k)}$, where for each $\boldsymbol{I}_n^{(k)} \in \mathbb{R}^{n\times n}$ all entries are zero, apart from position k, k which is one. This lets us state the full vectorized multi-channel graph convolution as

$$\begin{aligned} \text{vec}(\mathbf{W} * \boldsymbol{X}) &= (\boldsymbol{I}_n \otimes \boldsymbol{U})\left(\sum_{k=1}^{n} \hat{\mathbf{W}}_k \otimes \boldsymbol{I}_n^{(k)}\right)\left(\boldsymbol{I}_n \otimes \boldsymbol{U}^T\right)\text{vec}(\boldsymbol{X}) \\ &= \left(\sum_{k=1}^{n} \hat{\mathbf{W}}_k \otimes \boldsymbol{U}_{:,k}(\boldsymbol{U}_{:,k})^T\right)\text{vec}(\boldsymbol{X}) \end{aligned}$$

by using the fact that $\boldsymbol{U}\boldsymbol{I}_n^{(k)}\boldsymbol{U}^T = \boldsymbol{U}_{:,k}(\boldsymbol{U}_{:,k})^T$. We define $\boldsymbol{W}^{(k)} = \hat{\mathbf{W}}_k^T \in \mathbb{R}^{d\times c}$. Inverting the vec operation allows us to avoid the Kronecker product and state the exact

multi-channel graph convolution as

$$\boldsymbol{W} * \boldsymbol{X} = \sum_{k=1}^{n} \boldsymbol{U}_{:,k} (\boldsymbol{U}_{:,k})^T \boldsymbol{X} \boldsymbol{W}^{(k)} .$$

This concludes the proof.

Proof of Proposition 1.

Proof. We decompose $\boldsymbol{X}$ and $\boldsymbol{Y}$ as a sum of n rank-one matrices based on $\boldsymbol{U}$. We have

$$\boldsymbol{X} = \sum_{k=1}^{n} \boldsymbol{U}_{:,k} \boldsymbol{a}^{(k)} \in \mathbb{R}^{n \times d}$$

for $\boldsymbol{a}^{(k)} = (\boldsymbol{U}_{:,k})^T \boldsymbol{X} \in \mathbb{R}^{1 \times d}$ and

$$\boldsymbol{Y} = \sum_{k=1}^{n} \boldsymbol{U}_{:,k} \boldsymbol{b}^{(k)} \in \mathbb{R}^{n \times c}$$

for $\boldsymbol{b}^{(k)} = (\boldsymbol{U}_{:,k})^T \boldsymbol{Y} \in \mathbb{R}^{1 \times c}$. Thus,

$$\begin{aligned} \boldsymbol{\Theta} * \boldsymbol{X} &= \sum_{k=1}^{n} \boldsymbol{U}_{:,k} (\boldsymbol{U}_{:,k})^T \boldsymbol{U}_{:,k} \boldsymbol{a}^{(k)} \boldsymbol{W}^{(k)} \\ &= \sum_{k=1}^{n} \boldsymbol{U}_{:,k} \boldsymbol{a}^{(k)} \boldsymbol{W}^{(k)} \\ &= \sum_{k=1}^{n} \boldsymbol{U}_{:,k} \boldsymbol{b}^{(k)} = \boldsymbol{Y} \end{aligned} \tag{15}$$

for $(\boldsymbol{W}^{(k)})_{m,n} = \frac{b_n^{(k)}}{c \cdot a_m^{(k)}}$. By our assumption we have that $a_m^{(k)} \neq 0$ for all $k \in [n]$ and $m \in [d]$.

Proof of Proposition 2.

Proof. We utilize the eigendecomposition

$$\boldsymbol{A}^k = \sum_{j=1}^{n} \lambda_j^k \boldsymbol{U}_{:,j} (\boldsymbol{U}_{:,j})^T .$$

We then reformulate the polynomial filter as

$$\begin{aligned}\sum_{k=0}^{K} A^k \boldsymbol{X}\boldsymbol{V}^{(k)} &= \sum_{k=0}^{K}(\sum_{j=1}^{n} \lambda_i^k \boldsymbol{U}_{:,j}(\boldsymbol{U}_{:,j})^T)\boldsymbol{X}\boldsymbol{V}^{(k)} \\ &= \sum_{j=1}^{n} \boldsymbol{U}_{:,j}(\boldsymbol{U}_{:,j})^T \boldsymbol{X} \sum_{k=0}^{K} \lambda_i^k \boldsymbol{V}^{(k)} \\ &= \sum_{j=1}^{n} \boldsymbol{U}_{:,j}(\boldsymbol{U}_{:,j})^T \boldsymbol{X}\boldsymbol{W}^{(j)} \\ &= \boldsymbol{\Theta}_{\text{poly}} * \boldsymbol{X}\end{aligned} \tag{16}$$

where $\boldsymbol{W}_{(j)} = \sum_{k=0}^{K} \lambda_j^k \boldsymbol{V}^{(k)}$ and the corresponding $\boldsymbol{\Theta}_{\text{poly}}$

B.2 Proofs for Section 4

We now prove the injectivity and linear independence properties of the LMGC stated by Proposition 3 and Proposition 4. We first state and prove a helpful lemma:

Lemma 1. *Let $\mathcal{X}$ be a countable set, $K \geq 1$, and $\alpha_{(k)}^{(i,j)} \in \mathbb{R}$ a scalar for all $k \in [K], \boldsymbol{x}_{(i)}, \boldsymbol{x}_{(j)} \in \mathcal{X}$. Let all $\alpha_{(k)}^{(i,j)}$ be chosen such that for any two finite multisets $\mathcal{X}_1, \mathcal{X}_2 \subset \mathcal{X}$ and any $\boldsymbol{x}_p, \boldsymbol{x}_q \in \mathcal{X}$ with $\mathcal{X}_1 \neq \mathcal{X}_2$ or $\boldsymbol{x}_p \neq \boldsymbol{x}_q$, there exists a $k \in [K]$ such that $\sum_{\boldsymbol{x}_{(j)} \in \mathcal{X}_1} \alpha_{(k)}^{(p,j)} \boldsymbol{x}_{(j)} \neq \sum_{\boldsymbol{x}_{(j)} \in \mathcal{X}_2} \alpha_{(k)}^{(q,j)} \boldsymbol{x}_{(j)}$.*

Then, $\sum_{\boldsymbol{x}_{(j)} \in \mathcal{X}_p} \boldsymbol{W}_{(i,j)} \boldsymbol{x}_{(j)}$ is injective on finite multisets $\mathcal{X}_p \subset \mathcal{X}$ and elements $\boldsymbol{x}_i \in \mathcal{X}$ for a.e. $\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(K)} \in \mathbb{R}^{d \times c}$.

Proof. We want to show that for any two distinct pairs $\boldsymbol{x}_1 \in \mathcal{X}, \mathcal{X}_1 \subset \mathcal{X}$ and $\boldsymbol{x}_2 \in \mathcal{X}, \mathcal{X}_2 \subset \mathcal{X}$, their difference expressions

$$\sum_{\boldsymbol{x}_p \in \mathcal{X}_1} \boldsymbol{W}_{(1,p)} \boldsymbol{x}_p - \sum_{x_q \in \mathcal{X}_2} \boldsymbol{W}_{(2,q)} \boldsymbol{x}_q \neq \boldsymbol{0} \tag{17}$$

is nonzero. Substituting the definitions of $\boldsymbol{W}_{(1,p)}$ and $\boldsymbol{W}_{(2,q)}$, we obtain:

$$\sum_{k=1}^{K} \boldsymbol{W}^{(k)} (\sum_{\boldsymbol{x}_p \in \mathcal{X}_1} \alpha_{(k)}^{(1,p)} \boldsymbol{x}_p - \sum_{\boldsymbol{x}_q \in \mathcal{X}_2} \alpha_{(k)}^{(2,q)} \boldsymbol{x}_q) \neq \boldsymbol{0}\,. \tag{18}$$

For a.e. $\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(K)}$, this is zero only when all differences are zero. As such, we require all terms to be zero:

$$\begin{aligned}\sum_{\boldsymbol{x}_p \in \mathcal{X}_1} \alpha_{(1)}^{(1,p)} \boldsymbol{x}_p - \sum_{\boldsymbol{x}_q \in \mathcal{X}_2} \alpha_{(1)}^{(2,q)} \boldsymbol{x}_q \neq 0 \\ \wedge \cdots \wedge \sum_{\boldsymbol{x}_p \in \mathcal{X}_1} \alpha_{(K)}^{(1,p)} \boldsymbol{x}_p - \sum_{\boldsymbol{x}_q \in \mathcal{X}_2} \alpha_{(K)}^{(2,q)} \boldsymbol{x}_q \neq 0\,.\end{aligned} \tag{19}$$

As $\mathcal{X}$ is countable, a countable union of measure-zero sets has measure zero. This concludes the proof.

Proof of Proposition 3.

Proof. Continuing with the proof of Lemma 1, we need to show that

$$\sum_{\boldsymbol{x}_p \in \mathcal{X}_1} \alpha_{(k)}^{(1,p)} \boldsymbol{x}_p - \sum_{\boldsymbol{x}_q \in \mathcal{X}_2} \alpha_{(k)}^{(2,q)} \boldsymbol{x}_q \neq 0 \tag{20}$$

for some $k \in [K]$ for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ and $\mathcal{X}_1, \mathcal{X}_2 \subset \mathcal{X}$ with either $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ or $\mathcal{X}_1 \neq \mathcal{X}_2$. We can equivalently state Equation 20 in matrix notation $\boldsymbol{X}_s \boldsymbol{\alpha}_k \neq 0$ where the matrix

$$\boldsymbol{X}_s = \left[\boldsymbol{x}_{p_1} \ \dots \ \boldsymbol{x}_{p_{|\mathcal{X}_1|}} \ \boldsymbol{x}_{q_1} \ \dots \ \boldsymbol{x}_{q_{|\mathcal{X}_2|}} \right]$$

$\in \mathbb{R}^{d \times (|\mathcal{X}_1| + |\mathcal{X}_2|)}$ is defined as the concatenation of all elements in $\boldsymbol{x}_{p_1}, \dots, \boldsymbol{x}_{p_{|\mathcal{X}_1|}} \in \mathcal{X}_1$ and $\boldsymbol{x}_{q_1}, \dots, \boldsymbol{x}_{q_{|\mathcal{X}_2|}} \mathcal{X}_2$, and the vector $\boldsymbol{\alpha}_{(m)} = \left[\alpha_{(k)}^{(1,p_1)} \ \dots \ \alpha_{(k)}^{(1,p_{|\mathcal{X}_1|})} \ -\alpha_{(k)}^{(1,q_1)} \ \dots \ -\alpha_{(k)}^{(1,q_{|\mathcal{X}_2|})} \right]^T$ $\in \mathbb{R}^{|\mathcal{X}_1| + |\mathcal{X}_2|}$ for $k \in [K]$.

$\boldsymbol{X}_s \boldsymbol{\alpha}_k = 0$ is true if and only if $\boldsymbol{\alpha}_k \in \ker(\boldsymbol{X}_s)$. As $\ker(\boldsymbol{X}_s)$ spans a lower-dimensional subspace, and thus $\boldsymbol{X}_s \boldsymbol{\alpha}_k \neq 0$ is satisfied for a.e. choice of $\alpha_{(k)}^{(i,j)}$.

Proof of Proposition 4.

Proof. Similar to the proofs for Lemma 1 and Proposition 3 we need to show that for any element $\boldsymbol{x}_1 \boldsymbol{x}_2 \in \mathcal{X}$ and multisets $\mathcal{X}_1 \mathcal{X}_2 \subset \mathcal{X}$ with either $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ or $\mathcal{X}_1 \neq b \cdot \mathcal{X}_2$, we have

$$\sum_{\boldsymbol{x}_p \in \mathcal{X}_1} \boldsymbol{W}_{(1,p)} \boldsymbol{x}_p - c \cdot \sum_{\boldsymbol{x}_q \in \mathcal{X}_2} \boldsymbol{W}_{(2,q)} \boldsymbol{x}_q \neq 0 \tag{21}$$

for any $c \neq 0$ and a.e. $\alpha_{(k)}^{(i,j)}$ and a.e. $\boldsymbol{W}^{(k)}$ for all $k \in [K]$. This is equivalent to

$$\sum_{k=1}^{K} \boldsymbol{W}^{(k)} \left(\sum_{\boldsymbol{x}_p \in \mathcal{X}_1} \alpha_{(k)}^{(1,p)} \boldsymbol{x}_p - c \cdot \sum_{\boldsymbol{x}_q \in \mathcal{X}_2} \alpha_{(k)}^{(2,q)} \boldsymbol{x}_q \right) \neq 0 \tag{22}$$

We proceed with a proof by contradiction. Assume that the representations are linearly dependent. Then, there exist a constant $c^{(1,2)} \neq 0$, such that

$$\sum_{\boldsymbol{x}_p \in \mathcal{X}_1} \alpha_{(k)}^{(1,p)} \boldsymbol{x}_p - c^{(1,2)} \cdot \sum_{\boldsymbol{x}_q \in \mathcal{X}_2} \alpha_{(k)}^{(2,q)} \boldsymbol{x}_q = 0 \tag{23}$$

holds for all $k \in [K]$. This equality can only be satisfied for a.e. choice of coefficients $\alpha_{(k)}^{(i,j)}$ when

$$\sum_{\boldsymbol{x}_p \in \mathcal{X}_1} \boldsymbol{x}_p - c_k^{(1,2)} \cdot \sum_{\boldsymbol{x}_q \in \mathcal{X}_2} \boldsymbol{x}_q = 0 \tag{24}$$

for some $c_k^{(1,2)} \in \mathbb{R}$, i.e., the sum of the elements is linearly dependent. In this case, the value of $c^{(1,2)}$ is

$$c^{(1,2)} = \frac{\| \sum_{\boldsymbol{x}_p \in \mathcal{X}_1} \alpha_{(k)}^{(1,p)} \boldsymbol{x}_p \|}{\| \sum_{\boldsymbol{x}_q \in \mathcal{X}_2} \alpha_{(k)}^{(2,q)} \boldsymbol{x}_q \|} - \frac{(\| \sum_{\boldsymbol{x}_p \in \mathcal{X}_1} \boldsymbol{x}_p \| - c_k^{(1,2)} \cdot \| \sum_{\boldsymbol{x}_q \in \mathcal{X}_2} \boldsymbol{x}_q \|)}{\| \sum_{\boldsymbol{x}_q \in \mathcal{X}_2} \alpha_{(k)}^{(2,q)} \boldsymbol{x}_q \|} . \quad (25)$$

If $\mathcal{X}_1 = b \cdot \mathcal{X}_2$ for some scalar $b \neq 0$, then $c_k^{(1,2)} = 1$ for all $k \in [K]$, and thus also $c^{(1,2)} = 1$. As we assumed $\mathcal{X}_1 \neq b \cdot \mathcal{X}_2$, this case cannot occur. Under the assumption $\mathcal{X}_1 \neq b \cdot \mathcal{X}_2$, Equation 25 can only be satisfied for a single value of k simultaneously for a.e. $\alpha_{(k)}^{(i,j)}$. This contradicts the assumption of linear dependence. Thus, for $K > 1$, representations are linearly independent.

C Experimental Details

All experiments were executed on an Nvidia H100 GPU with 96GB memory.

C.1 Model Architectures

Graph-Level Prediction We evaluate a model that first applies a linear feature encoder that maps the feature dimension onto a hidden dimension d. Then, k iterations of message-passing are applied, each followed by a GELU activation. For cases with residual connections, the input to each message-passing layer is added to the output of the GELU activation. This output is stored for all message-passing steps for jumping knowledge. After message-passing, the channel-wise maximum value per node is used when jumping knowledge is allowed. Otherwise, the output of the last layer is used. As a decoder, we use a two-layer MLP with a linear layer followed by a GELU activation and another linear layer.

Node Classification The employed model first applies a linear encoder, ReLU activation, and a dropout layer. Then, k layers of message-passing with a residual connection are applied, each followed by a ReLU activation and a dropout layer. As the last operation, a single linear layer is applied. The number of channels for all layers is set to a shared d, apart from the initial and final dimensions. For these experiments, k is set to 2. The number of channels is reduced so that the total number of parameters is maximal but below $100\,000$.

C.2 Hyperparameters

For the experiments in Section 6.2, we optimize the learning rate in $\{0.001, 0.0003, 0.0001\}$ and the number of layers in $\{6, 8, 10\}$ using a grid search. Each experiment is repeated

for four fixed seeds. Average train and test loss are reported. Optimal hyperparameters are shown in Table 7 and Table 4.

For the experiments in Section 6.3, we optimize the learning rate in $\{0.01, 0.003, 0.001\}$ and the dropout ratio in $\{0.0, 0.25, 0.5\}$ using a grid search. Each experiment is run for ten splits into train, validation and test sets. Based on the best accuracy on the validation set, we reuse the optimal hyperparameters and run all ten splits for five times and report average test scores. Optimal hyperparameters are shown in Table 5.

C.3 Additional Results

We provide additional results on the full ZINC dataset that contains around $250\,000$ molecular graphs, training errors, and runtimes. Average train and test errors are presented in Table 6. Similarly to the experiment on universality, the LMGC achieves the lowest training error. The difference is more significant on the ZINC12k subset as it is easier to overfit on less data. This improvement also leads to the LMGC having the lowest test loss for ZINC and ZINC12k. Compared to GATv2 and FAGCN, the LMGC improves the test results by at least 40%. This confirms the ability of the LMGC to match the expressivity of GIN. The training time per epoch of the LMGC is increased compared to GIN by around 35% but slightly reduced compared to GATv2 and FAGCN as the LMGC does not require the normalization step.