Preventing Representational Rank Collapse in MPNNs by Splitting the Computational Graph

Andreas Roth[1] **Franka Bause**[2,3] **Nils M. Kriege**[2,4] **Thomas Liebig**[1,5]
[1]Faculty of Computer Science, TU Dortmund University, Dortmund, Germany
{andreas.roth, thomas.liebig}@tu-dortmund.de
[2]Faculty of Computer Science, University of Vienna, Vienna, Austria
[3]UniVie Doctoral School Computer Science, University of Vienna, Vienna, Austria
[4]Research Network Data Science, University of Vienna, Vienna, Austria
{franka.bause, nils.kriege}@univie.ac.at
[5]Lamarr Institute for Machine Learning and Artificial Intelligence, Dortmund, Germany

Abstract

The ability of message-passing neural networks (MPNNs) to fit complex functions over graphs is limited as most graph convolutions amplify the same signal across all feature channels, a phenomenon known as rank collapse, and over-smoothing as a special case. Most approaches to mitigate over-smoothing extend common message-passing schemes, e.g., the graph convolutional network, by utilizing residual connections, gating mechanisms, normalization, or regularization techniques. Our work contrarily proposes to directly tackle the cause of this issue by modifying the message-passing scheme and exchanging different types of messages using multi-relational graphs. We identify a sufficient condition to ensure linearly independent node representations. As one instantion, we show that operating on multiple directed acyclic graphs always satisfies our condition and propose to obtain these by defining a strict partial ordering of the nodes. We conduct comprehensive experiments that confirm the benefits of operating on multi-relational graphs to achieve more informative node representations.

1 Introduction

Many challenging tasks, such as drug discovery [1], social network predictions [2], and traffic prediction [3], involve graph-structured data. Message-passing neural networks (MPNNs) [4] have found success in many of these areas. However, MPNNs did not see the same level of improvement against classical methods, such as graph kernels [5], that was achieved for computer vision [6] and natural language processing tasks [7]. MPNNs struggle to achieve satisfying performance for challenging tasks, such as large-scale heterophilic node classification. Computational issues like over-smoothing [8] and its more general form of representational rank collapse limit the ability of MPNNs to obtain informative node embeddings. Rank collapse refers to the phenomenon that node representations become more similar in each feature dimension after each iteration of message-passing. Much attention has been paid to dealing with over-smoothing, e.g., by allowing either smooth signals or non-smooth signals to be amplified [9]. However, as MPNNs typically produce multi-dimensional features, models should be able to produce representations that are similar in one dimension but dissimilar in another dimension, e.g., when two people work together but pursue different hobbies. Few methods have yet to consider capturing both smooth and non-smooth features simultaneously.

In particular, it was identified that applying graph convolutions using a simple graph does not allow a different behavior across features [10, 11]. In this work, we propose splitting a given graph into a multi-relational graph and operating on multi-relational split MPNNs (MRS-MPNNs). Each edge is assigned a relation type, and messages are passed using distinct feature transformations before the

Roth et al., Preventing Representational Rank Collapse in MPNNs by Splitting the Computational Graph. *Proceedings of the Third Learning on Graphs Conference (LoG 2024)*, PMLR 269, Virtual Event, November 26–29, 2024.

results are combined to form a single state within each graph convolution. Our theoretical analysis provides sufficient conditions to ensure more informative node representations. Specifically, different structural properties between edge relations are required. While this theory opens many different directions for constructing multiple graphs that satisfy our conditions, we propose an instantiation that forms directed acyclic graphs (DAGs) with each relation type. These are constructed by defining a strict partial ordering over the nodes and assigning each edge to a relation type depending on the ordering between the two nodes. In our experiments, we evaluate several choices for the partial order and identify the node degree as a powerful and general choice. We empirically confirm that MRS-MPNN prevents rank collapse and that its ability to amplify different signals for each feature column benefits the learning process for several methods. We summarize our main contributions as follows:

- We propose to split the edges of graphs in multiple edge relations and utilize multi-relational split MPNNs (MRS-MPNNs). We establish the necessary and sufficient condition on the edges of each relation type so that node representations become more informative (Section 4).
- As one instantiation that satisfies our condition, we propose to utilize multiple edge relations that are directed and acyclic for which define a strict partial ordering of the nodes. Each edge is assigned a relation type according to the ordering of its adjacent nodes (Section 5).
- Our experiments confirm our theory by demonstrating that MRS-MPNNs prevent rank collapse and improve the learning process (Section 6).

2 Preliminaries

Let $G = (\mathcal{V}, \mathcal{E})$ be a (simple) graph, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ is its set of n nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ its set of edges. We refer to $\mathbf{A} \in \mathbb{R}^{n \times n}$ as the adjacency matrix for which $\mathbf{A}_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$, otherwise the entry is 0. The nodes with an edge ending at v_i are defined as its neighbors $N_i = \{v_k \mid (v_k, v_i) \in \mathcal{E}\}$. Based on each node's incoming edges, we define the degree matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ as a diagonal matrix with $d_{ii} = |N_i|$. A multi-relational graph $(\mathcal{V}, \mathcal{E}_1, \ldots, \mathcal{E}_l)$ is a graph that contains multiple relations or edge types.

Message-Passing Neural Networks. Given a graph G and d-dimensional features $\mathbf{X} \in \mathbb{R}^{n \times d}$ for each node, message-passing neural networks (MPNNs) aim to obtain informative node representations capturing both structural properties and connections between node features. Most MPNNs follow an iterative message-passing scheme that updates each node's representations

$$\mathbf{x}_i' = \phi\left(\mathbf{x}_i, \bigoplus_{j \in N_i} \psi\left(\mathbf{x}_i, \mathbf{x}_j\right)\right) \tag{1}$$

using a message function ψ, a permutation invariant aggregation function $\oplus$, and a combine function ϕ. This is typically repeated for k instantiations. State-of-the-art MPNNs add additional components such as residual connections [12, 13], restart terms [12, 14], or gating mechanisms [15, 16]. However, for exchanging messages, most methods follow a simple scheme that can be expressed in matrix notation

$$\bigoplus_{j \in N_i} \psi\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left[\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}\right]_i \tag{2}$$

where $\tilde{\mathbf{A}} \in \mathbb{R}^{n \times n}$ corresponds to the aggregation function. $\tilde{\mathbf{A}}$ may be the symmetrically normalized adjacency matrix, the mean aggregation, the sum aggregation, or contain negative values. It may also include self-loops. Most models apply a linear feature transformation $\mathbf{W} \in \mathbb{R}^{d \times d'}$.

Rank Collapse and Over-Smoothing. For many MPNNs, node representations tend to become more similar as more update iterations are performed, limiting the learnable functions over graphs. All graph convolutions of the form given by Eq. 2 were found to amplify and damp the same signals across all features [10, 11]. While the features for two adjacent nodes can get closer to each other or further apart, this behavior is the same for all features. In the limit, all node representations become linearly dependent, resulting in a rank-one matrix. This phenomenon limits the information that can be present in representations and is referred to as rank collapse [10]. As a special case of rank collapse, over-smoothing occurs when representations become linearly dependent and contain smooth values [8, 17–19].

3 Related Work

Dealing with Rank Collapse and Over-Smoothing. Various methods aim to reduce over-smoothing in MPNNs. These include combining the output of Eq. 2 with previous states, e.g., by utilizing residual connections [12, 13, 20] or restart terms [12, 14, 21]. Gating mechanisms were proposed to stop updating node states after varying numbers of iterations [16, 22]. Normalization operations that reduce the similarity between representations were introduced [23, 24]. Another line of research proposes regularization terms to punish smooth representations during optimization [25].

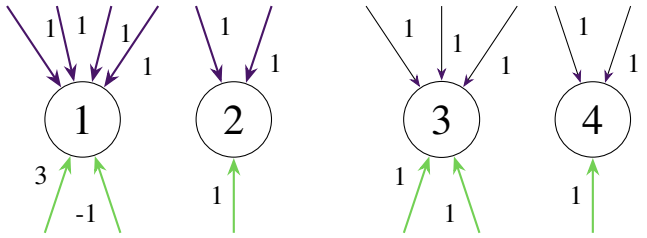
Several works study the construction of MPNNs that can amplify different signals, e.g., either low-frequency or high-frequency signals [19, 26], or mixtures using negative edge weights [9, 27] or combining multiple aggregation functions [28, 29]. It is still unclear how a single graph convolution can amplify different signals for each feature channel, as few methods have been proposed to mitigate the more general rank collapse. Jin et al. [30] propose to regularize the correlation between node representations. All of these methods utilize the base message-passing scheme (Eq. 2) using a single edge relation, which is known to suffer from rank collapse for almost every $\tilde{\mathbf{A}}$ and $\mathbf{W}$ [10, 11].

Changing the Computational Graphs. Typically, MPNNs operate directly on the input graph $G = (\mathcal{V}, \mathcal{E})$, which may cause computational issues like over-squashing [31] as information cannot flow over large distances. Several methods propose to perform the message-passing on a computational graph $G' = (\mathcal{V}, \mathcal{E}')$ that has better-suited structural properties, e.g., by graph rewiring techniques [32–34]. Co-GNNs modify the computational graph in each step by allowing each node to choose between sending messages, listening, or isolating [22]. Dense connectivity, including higher node degrees [9] and a larger curvature [35] were identified to amplify over-smoothing. However, structural properties leading to rank collapse and how to construct beneficial computational graphs remain unclear.

Various MPNNs for multi-relational graphs have been proposed [36, 37]. However, these assume multiple relations to be given in the data. Other methods explored forming multiple computational graphs. Suresh et al. [38] add computational graphs that connect nodes with a large structural similarity to account for both proximity and structural similarity with different edge types. Factorizable graph convolutional networks disentangle edges of a graph into multiple interpretable factor graphs to produce disentangled representations [39]. ES-GNN [40] learns to split the edges of a graph into two sets, one containing task-relevant edges, and one containing task-irrelevant edges. Predictions are obtained by performing message-passing using the task-relevant edges. EXPASS [41] modifies the computational graph by weighting edges so that edge weights of low importance based on an explanation method are reduced. ACM [42] uses both the graph Laplacian and normalized adjacency matrices as computational graphs to amplify low-frequency and high-frequency signals. ADR-GNNs [43] learn an advection-diffusion-reaction system for which edge weights are learned channel-wise independently, but a shared transformation is applied. For directed graphs, Dir-GNNs add the reverse direction as a second computational graph [44]. Subgraph GNNs [45, 46] similarly generate multiple subgraphs based on a given policy. Message-passing is performed separately for each subgraph, which can thus lead to representational rank collapse on each subgraph. Despite these works, the general benefits of operating on multiple computational graphs and how these lead to more informative node representations are still unclear. Studying the potential benefits and required properties of multiple computational graphs can help us better understand the advantages of all these methods.

4 Splitting MPNNs into Multi-Relational MPNNs

As each iteration of message-passing on a simple graph makes feature columns more similar [10, 11], we study the effects of utilizing multiple edge relations by splitting a graph into a multi-relational graph. The goal is to be able to obtain more informative embeddings by allowing for features to become more similar in one dimension while becoming more dissimilar in another dimension. Formally, we let l be the number of relation types we want to split our graph into. For each edge $(v_i, v_j) \in \mathcal{E}$, we apply a permutation invariant edge relation assignment function $f\colon (v_i, v_j) \mapsto \{1, \dots, l\}$. Note that while we consider assigning each edge to a single relation type, multiple assignments and continuous assignment scores are also covered by our following theory. Based on this assignment function, one of the relation types and corresponding feature transformations $\psi_1, \dots, \psi_l$ is selected for each edge. In general, we define one iteration of MRS-MPNNs as follows:



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

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

(c) Numerical dependency.

Figure 1: Structurally dependent (a) and independent (b) node pairs. Colors indicate different graphs. Numbers beside edges indicate edge weights. SD refers to the structural dependency of nodes based on d_1 and d_2 (given by Def. 4.2).

Definition 4.1. (Multi-Relational Split MPNNs (MRS-MPNNs))

$$\mathbf{x}_i' = \phi\left(\mathbf{x}_i, \bigoplus_{j \in N_i} \psi_{f(v_i,v_j)}\left(\mathbf{x}_i, \mathbf{x}_j\right)\right), \tag{3}$$

where $\bigoplus$ and ϕ are an aggregation function and a combination function as used in an MPNN.

Thus, all MPNNs can be transformed into an MRS-MPNN by duplicating the message function and defining an edge relation assignment function f. As an example, we state the graph convolutional network (GCN) [47] in the MRS framework, as it is commonly used and often serves as the message-passing component within complex models.

MRS-GCN. Given some node representations $\mathbf{X}^{n\times d}$, we define the MRS-GCN as

$$\begin{aligned}\left[\text{MRS-GCN}\left(\mathbf{X}, \mathcal{E}, f\right)\right]_i &\coloneqq \left[\tilde{\mathbf{A}}_1\mathbf{X}\mathbf{W}_1 + \cdots + \tilde{\mathbf{A}}_l\mathbf{X}\mathbf{W}_l\right]_i \\ &= \sum_{j\in N_i} \frac{1}{\sqrt{d_i}\sqrt{d_j}}\mathbf{W}_{f(v_i,v_j)}\mathbf{x}_j,\end{aligned} \tag{4}$$

where d_i is the degree of node i, $\mathbf{W}_1, \ldots, \mathbf{W}_l \in \mathbb{R}^{d\times d'}$ are feature transformations and $\tilde{\mathbf{A}}_1, \ldots, \tilde{\mathbf{A}}_l \in \mathbb{R}^{n\times n}$ contain the edge weights of the corresponding computational graph, i.e., $\tilde{\mathbf{A}}_1 + \cdots + \tilde{\mathbf{A}}_3 = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. Compared to the GCN, only the selected transformation $\mathbf{W}_{f(v_i,v_j)}$ changes. We define additional models in their MRS form in Appendix B.1.

Theoretical Properties. We now show the computational benefits of operating on multiple relations. We do that by identifying a sufficient condition on the structure of the edge sets, that ensures linear independence of node representations. In this analysis, we consider instantiations of MRS-MPNN of the form

$$\mathbf{X}' = \sigma(\mathbf{A}_1\mathbf{X}\mathbf{W}_1 + \cdots + \mathbf{A}_l\mathbf{X}\mathbf{W}_l) \tag{5}$$

where $\mathbf{X} \in \mathbb{R}^{n\times d}$ are node representations, $\mathbf{A}_1, \ldots, \mathbf{A}_l \in \mathbb{R}^{n\times n}$ represent computational graphs, and $\mathbf{W}_1, \ldots, \mathbf{W}_l \in \mathbb{R}^{d\times d'}$ are feature transformations. Note, that MRS-GCN and MRS-SAGE are special cases of this form. Our analysis requires the weighted in-degree of each node and graph:

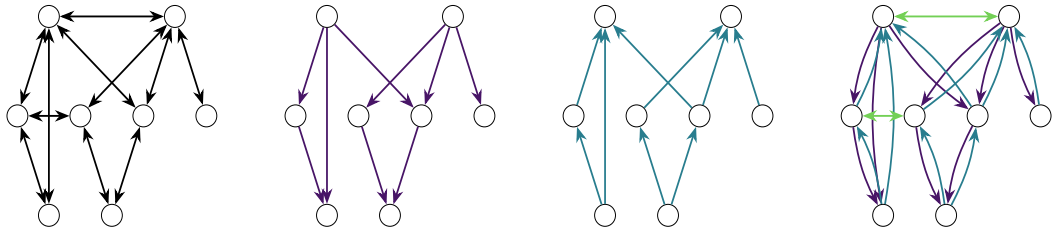
Definition 4.2. (Weighted in-degrees) Let $\mathbf{A}_1, \ldots, \mathbf{A}_l \in \mathbb{R}^{n\times n}$ represent l edge relations with any edge weights. For each node $i \in [n]$, the vector of weighted in-degrees is defined as

$$\mathbf{d}^i = \begin{bmatrix} d_1^i & \ldots & d_l^i \end{bmatrix}, \tag{6}$$

where $d_k^i = \sum_{m\in[n]} \mathbf{A}_k[i,m]$ is the weighted in-degree of node i in edge relation k.

With this, we introduce the concept of structural dependence of a pair of nodes as the linear dependence of their weighted in-degrees:

Definition 4.3. (Structural dependence and independence) Let $\mathbf{A}_1, \ldots, \mathbf{A}_l \in \mathbb{R}^{n\times n}$ be matrices and $\mathbf{d}^i = \begin{bmatrix} d_1^i & \ldots & d_l^i \end{bmatrix} \in \mathbb{R}^l$ be the vector of weighted in-degrees for $i \in [n]$. A pair of nodes v_i, v_j is said to be **structurally independent** if the vectors $\mathbf{d}^i$ and $\mathbf{d}^j$ are linearly independent. Otherwise, they are called **structurally dependent**.



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

Figure 2: Different computational graphs for MRS-MPNN$_{\text{DA}}$.

We provide an example in Figure 1. This serves as a sufficient condition that ensures that node pairs get mapped to linearly independent representations:

Theorem 4.4. *(Structurally independent nodes produce linearly independent representations.) Let* $\mathbf{A}_1, \ldots, \mathbf{A}_l \in \mathbb{R}^{n \times n}$ *be* l *matrices with nodes* v_i, v_j *being structurally independent. Then,*

$$\mathbf{x}'_i = [\mathbf{A}_1\mathbf{X}\mathbf{W}_1 + \cdots + \mathbf{A}_l\mathbf{X}\mathbf{W}_l]_i\,, \tag{7}$$

$$\mathbf{x}'_j = [\mathbf{A}_1\mathbf{X}\mathbf{W}_1 + \cdots + \mathbf{A}_l\mathbf{X}\mathbf{W}_l]_j\,, \tag{8}$$

are linearly independent for a.e. $\mathbf{W}_1, \ldots, \mathbf{W}_l \in \mathbb{R}^{d \times d'}$ *with* $d, d' > l \geq 1$ *and a.e.* $\mathbf{X} \in \mathbb{R}^{n \times d}$ *with* $\operatorname{rank}(\mathbf{X}) = 1$

We provide all proofs in Appendix A. Linearly dependent node representations do not form fixed points or invariant subspaces for two structurally independent nodes. This makes it impossible for two such nodes to converge to overly similar representations. Node representations can get more similar in one feature dimension and more dissimilar in another dimension, allowing representations to capture more information. Also note that for component-wise injective activation functions σ, we also know that $\sigma(\mathbf{x}_i)$ is linearly independent to $\sigma(\mathbf{x}_j)$ for a.e. vectors $\mathbf{x}_i, \mathbf{x}_j$. Another intuitive explanation comes from connecting structurally independent nodes to their unfolding trees [48]: These nodes always have different unfolding trees, and MRS-MPNNs will assign not just different but linearly independent representations.

Consequently, rank collapse, and therefore over-smoothing, are prevented when operating on multiple graphs with structurally independent nodes. As a side note, we also highlight the connection to the expressivity of an MPNN. Both phenomena have the goal of preventing node states from becoming equal when their inputs are different. Two structurally independent nodes always get distinguished. Thus, the more structurally independent nodes exist in a graph, the more nodes are distinguished.

We are interested not only in this structural expressivity but also in the information stored in the node representations.

Theorem 4.5. *(Structural independence prevents rank collapse.) Given* n *nodes, let* $\mathbf{E} = \begin{bmatrix}\mathbf{d}^1 & \ldots & \mathbf{d}^n\end{bmatrix} \in \mathbb{R}^{n \times l}$ *be the matrix of weighted in-degrees. Let* σ *be a component-wise injective activation function. Then, for a.e.* $\mathbf{W}_1, \ldots \mathbf{W}_l \in \mathbf{R}^{d \times d'}$ *with* $n \geq d' \geq l$*,*

$$\operatorname{rank}(\sigma(\sum_{l=1}^{k} \mathbf{A}_l\mathbf{X}\mathbf{W}_l)) \geq l, \tag{9}$$

where $l = \operatorname{rank}(\mathbf{E})$ *for a.e.* $\mathbf{X} \in \mathbb{R}^{n \times d}$.

We have now confirmed that the minimal rank of representations gets larger with more structurally independent nodes.

5 Obtaining Multiple Relations using a Partial Ordering

A suitable edge relation assignment function $f\colon (v_i, v_j) \mapsto \{1, \ldots, l\}$ can be constructed in various ways. These include application-specific graph assignments, e.g., different relations for specific atom pairs in molecular graphs, data-dependent assignments, e.g., depending on initial or intermediate features, or assignments based on structural graph properties.

We derive one method based on observing that attentional message-passing, e.g., graph attention network [49] or graph transformers [50], utilize multiple attention heads, which can be seen as multiple edge relation type. For each attention head or relation type, the attention scores are normalized using the softmax function, i.e., each node aggregates its neighbors using a weighted mean for aggregation on each relation. These are typically applied to ergodic graphs or relations, which refer to them being strongly connected and aperiodic, i.e., a path exists between each pair of nodes, and the lengths of its cycles do not have a common divisor > 1. This is a typical assumption that is also used for theoretical studies on over-smoothing [10, 17, 18].

Corollary 5.1. *Let $\mathbf{A}_1,\dots,\mathbf{A}_l$ represent ergodic relations with a weighted mean aggregation. Then, there are no structurally independent nodes.*

However, graphs without ergodic components behave differently. Graphs that do not contain any ergodic parts are directed acyclic graphs (DAGs), which we formally define as follows:

Definition 5.2. (DAG) A graph $G = (\mathcal{V}, \mathcal{E})$ is a directed acyclic graph if there exists a strict partial ordering $\prec$ on the nodes such that all edges $(v_i, v_j) \in [n] \times [n]$ satisfy $v_i \prec v_j$.

We further refer to nodes without incoming edges as root nodes and to edge relations $\mathbf{E}$ as directed acyclic relations (DARs) if their edges induce a DAG. Utilizing multiple DARs ensures that structurally independent nodes always exist.

Proposition 5.3. *Let $\mathcal{E}_1, \mathcal{E}_2$ represent two non-empty DARs with different root nodes. Then, there always exist structurally independent nodes for any non-zero edge weights.*

Thus, constructing multiple DARs is one way to benefit from multiple relations that will result in non-smooth representation even when utilizing the mean as the aggregation function. Transforming a given edge set into a DAR is a well-known algorithmic challenge in graph theory, which is connected to topological sorting [51] and finding feedback arc sets [52]. In this work, we transform any graph into DARs by defining a strict partial ordering $\prec$ on the nodes. The first DAR is obtained by filtering the edges (i, j) to only consider those for which $v_i \prec v_j$, i.e., $\mathcal{E}_1 = \{(v_i, v_j) \in \mathcal{E} \mid v_i \prec v_j\}$. The converse ordering provides a second DAR, i.e., $\mathcal{E}_2 = \{(v_i, v_j) \in \mathcal{E} \mid v_j \prec v_i\}$. For some edges, neither $v_i \prec v_j$ nor $v_j \prec v_i$ may be satisfied. These edges are not contained in $\mathcal{E}_1$ or $\mathcal{E}_2$. To retain all edges, we propose a third relation type that contains all the remaining edges, i.e., $\mathcal{E}_3 = \mathcal{E} \setminus (\mathcal{E}_1 \cup \mathcal{E}_2)$. Based on this, we define the graph assignment function to be

$$f(v_i, v_j) = \begin{cases} 1, & \text{if } v_i \prec v_j \\ 2, & \text{if } v_j \prec v_i \\ 3, & \text{otherwise} \end{cases} \tag{10}$$

We provide an example of such a multi-relational graph in Figure 2. We refer to an MRS-MPNN instantiation that uses this relation assignment function as MRS-MPNN$_{\text{DA}}$. Note that additional graphs cannot reduce the minimal rank of representations. In fact, when $\mathcal{E}_3$ does not have a regularity structure and leads to further structural independent nodes, we can ensure that the minimal rank is at least 3.

Many graph theoretical algorithms, such as graph traversal algorithms or centrality measures like the node degree, can provide a strict partial ordering for nodes. While any partial ordering allows for more informative node representations, the three computational graphs should benefit from different feature transformations. Messages within each computational graph should be more similar to each other than to messages of other computational graphs. For example, when choosing the node degree as our partial ordering, messages sent from higher-degree nodes to lower-degree nodes differ from those sent from lower-degree nodes to higher-degree nodes. The third relation constructs messages between nodes of the same degree. The task should then benefit from different message types between these nodes. In molecular data, higher-degree nodes correspond to other atoms than lower-degree nodes [53], which may benefit from sending different message types. As this choice has a large effect on the MRS-MPNN, this allows future models to benefit from application-dependent domain knowledge. Many other splitting approaches that do not construct DARs can work similarly well as long as these contain structurally independent nodes.

Table 1: Mean values and standard deviations over three runs on the ZINC12k dataset (best results marked in **bold**). The learning rate and the number of layers are tuned. Train MAE is the overall minimum, while test MAE is based on the best validation MAE. Step times are in milliseconds (ms).

Method	Step Time (ms)	ZINC12k (MAE) Train	ZINC12k (MAE) Test
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	$\mathbf{0.003} \pm 0.001$	$\mathbf{0.318} \pm 0.031$

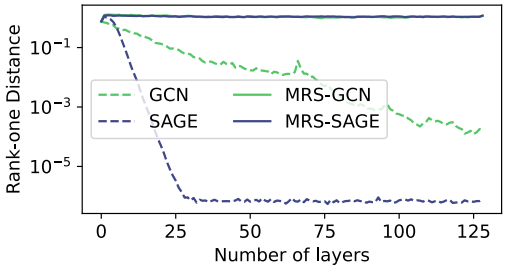


Figure 3: Comparison of the Rank-one distance (ROD). Mean values over 50 random seeds.

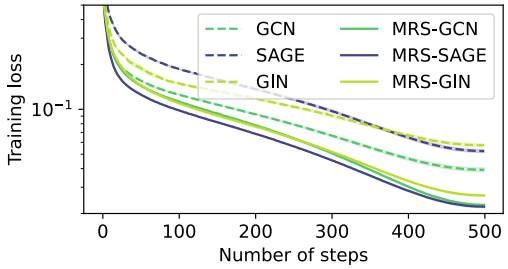


Figure 4: Training loss (MAE) during optimization on ZINC. The learning rate and the number of layers are tuned for each MPNN. Mean values over three runs with standard deviations as semi-transparent areas.

6 Experiments

We now investigate the ability of MRS-MPNNs to improve learning for complex tasks. We provide additional details in Appendix B and reproducible code as supplementary material[1].

6.1 Improving the Learning Process

We first consider the challenging ZINC dataset [54]. ZINC is a single-label graph regression task. For some of our ablation studies we use the subset containing 12 000 graphs (ZINC12k), as proposed by Dwivedi et al. [55]. Following Dwivedi et al. [56], all models use at most 500 000 parameters. AdamW [57] is used for optimization. We integrate our models into the implementation of Tönshoff et al. [58].

Evaluating Node Orderings. We evaluate several strategies for constructing node orderings using the ZINC12k dataset. We consider random values, the sum of initial node features, Personalized PageRank (PPR) scores, and the node degree. The strategies are evaluated in terms of execution time per training step, minimal achieved training loss, and test loss corresponding to the best validation performance. We tune the learning rate and the number of layers for each method. All orderings are applied to MRS-GCN$_{\text{DA}}$.

Results are presented in Table 1. While the GCN underfits the training data, all MRS-GCNs significantly improve the training loss. The random ordering worsens the test performance. This highlights the importance of constructing relations that align with the given data and task. For the other orderings, the test performance is improved as a consequence of the improved training loss. The degree-based ordering achieves the best train and test performance. Runtime is increased by around 35% using our implementation. A detailed runtime evaluation is provided in Table 6. The notable differences in performance between orderings show potential for optimal task-dependent

[1] Code is available at https://github.com/roth-andreas/splitting-computational-graphs

Table 2: Mean values and standard deviations over three runs on the ZINC dataset (best results marked in **bold**). The learning rate and the number of layers are tuned. Train MAE is the overall minimum, while test MAE is based on the best validation MAE. Step times are in milliseconds (ms).

METHOD	ZINC (MAE)	
	TRAIN	TEST
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$_{\text{DA}}$	$\mathbf{0.011} \pm 0.003$	$\mathbf{0.088} \pm 0.004$

Table 3: Test results on ZINC12k. The base models are combined with residual connections (Res), jumping knowledge (JK), and Laplacian positional encoding (LapPE). The learning rate is tuned for each entry. Mean average error (MAE) over three runs is reported (pairwise best results marked in **bold**). Standard deviations are shown in Table 11.

Method	1	2	4	8	16	32
GCN	0.591	0.493	0.421	0.404	0.417	0.440
MRS-GCN$_{\text{DA}}$	**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$_{\text{DA}}$ + 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$_{\text{DA}}$ + 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$_{\text{DA}}$ + LapPE	**0.441**	**0.363**	**0.292**	**0.272**	**0.297**	**0.317**

orderings and graph splittings. To study further properties of multi-relational MPNNs, we utilize the node degree as our ordering for all other experiments.

More Informative Node Representations. To empirically validate that the representations obtained by MRS-MPNNs can be more informative than those obtained by MPNNs, we compare their distance to a rank-one matrix throughout 128 iterations using the rank-one distance (ROD) [11]. In each iteration, one message-passing step and the ReLU activation function are applied, and ROD is calculated. We repeat this process for 50 random graphs from the ZINC dataset. Results for GCN, MRS-GCN$_{\text{DA}}$, SAGE, and MRS-SAGE$_{\text{DA}}$ are shown in Figure 3. The representations for GCN and SAGE converge to a rank-one state, while the distance to a rank-one matrix remains roughly constant across all 128 layers for the DA versions.

Additional Experiments. To better understand the advantages of MRS-MPNNs, we present training and test performances for ZINC using the GCN, SAGE, GAT [49], GIN [59], GatedGCN [55], and their corresponding MRS-MPNNs in Table 2. We provide details about each model in Appendix B.1. In all cases, the respective MRS-MPNNs achieve a significantly reduced training loss. Due to the ability of MRS-MPNNs to amplify multiple signals, they suffer much less from underfitting and consequently improve the test performance. To emphasize this, we display the training loss during optimization in Figure 4. MRS-MPNNs visibly improve the optimization process.

GCNs are often combined with more advanced techniques, like residual connections (Res) [6], jumping knowledge (JK) [60], and Laplacian positional encodings [61]. We evaluate their interplay

Table 4: Mean accuracy and standard deviation for five directed benchmark graphs (best result marked in **bold** and 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-GCN$_{\text{DA}}$	$\mathbf{76.01} \pm 1.90$	$\mathbf{80.17} \pm 1.88$	$\mathbf{66.03} \pm 0.20$	$\mathbf{74.72} \pm 0.05$	$\mathbf{91.87} \pm 0.42$

with MRS-MPNNs for various layers and present test scores in Table 3. MRS-GCNs outperform their corresponding GCNs in all cases, and achieve their best performance with deeper models. We note a slight drop in performance for 32 layers in some cases for MRS-GCNs. This can be attributed to optimization challenges, i.e., exploding and vanishing gradients, as normalization layers were not considered here.

6.2 Comparison with State-of-the-Art

Improved optimization of the target function suggests that MRS-MPNNs will be particularly valuable for challenging tasks. We consider large-scale heterophilic graphs, namely Squirrel and Chameleon [62], Arxiv-Year and Snap-Patents [63], and Roman-Empire [64]. Our implementation is based on Dir-GNN [44], a state-of-the-art method. We replace the MPNN modules of Dir-GNN with the corresponding MRS-MPNN$_{\text{DA}}$ modules. Accordingly, MRS-SAGE$_{\text{DA}}$ is used for Roman-Empire, while MRS-GCN$_{\text{DA}}$ is used for the other datasets. Experiments for Squirrel, Chameleon, and Roman-Empire are repeated for ten fixed splits into training, validation, and test sets and for Arxiv-Year and Snap-Patents for five fixed splits. The baseline results are reused from our reference implementation by Rossi et al. [44]. Based on their hyperparameter values, we tune the learning rate, number of layers, and dropout ratio using a grid search. We compare to seven state-of-the-art methods for heterophilic graphs and three for directed graphs. Further experimental details are provided in Appendix B.

We present the test results in Table 4. MRS-MPNNs slightly improve the performance for all five datasets, with more significant gains for the larger datasets, i.e., Arxiv-Year and Snap-Patents. As with our other experiments, we observe larger improvements in the training loss.

7 Conclusion

In this work, we propose to split graphs into multi-relational graphs and operate MPNNs on these. We identify the necessary and sufficient condition on these relations that ensures that representations will always have more linearly independent features. While we show that this is always satisfied when operating on multiple DARs, many other graph splitting techniques can be designed based on our theory. Our experiments confirm that operating with multiple relations results in more informative node representations, which improves learning. As limitations of MR-MPNNs, we have seen an increase in runtime, the need to find a suitable graph splitting method and that they tend to overfit on the training data. We anticipate further opportunities for other graph splitting methods that satisfy our condition. Task-specific knowledge and invariances for graph splitting can provide additional benefits.

Acknowledgements

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. This work was supported by the Vienna Science and Technology Fund (WWTF) [10.47379/VRG19009]. Simulations were performed with computing resources granted by WestAI under project rwth1631.

References

[1] Ilia Igashov, Hannes Stärk, Clément Vignac, Arne Schneuing, Victor Garcia Satorras, Pascal Frossard, Max Welling, Michael M. Bronstein, and Bruno E. Correia. Equivariant 3d-conditional diffusion model for molecular linker design. *Nat. Mac. Intell.*, 6(4):417–427, 2024. doi: 10.1038/S42256-024-00815-9. 1

[2] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia, editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 417–426. ACM, 2019. doi: 10.1145/3308558.3313488. 1

[3] Austin Derrow-Pinion, Jennifer She, David Wong, Oliver Lange, Todd Hester, Luis Perez, Marc Nunkesser, Seongjae Lee, Xueying Guo, Brett Wiltshire, Peter W. Battaglia, Vishal Gupta, Ang Li, Zhongwen Xu, Alvaro Sanchez-Gonzalez, Yujia Li, and Petar Velickovic. ETA prediction with graph neural networks in google maps. In Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong, editors, *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pages 3767–3776. ACM, 2021. doi: 10.1145/3459637.3481916. 1

[4] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017. 1

[5] Nils M. Kriege. Weisfeiler and leman go walking: Random walk kernels revisited. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *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*, 2022. 1

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 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*, pages 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. 1, 8

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. 1

[8] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *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*, pages 3538–3545. AAAI Press, 2018. doi: 10.1609/AAAI.V32I1.11604. 1, 2

[9] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In

Xingquan Zhu, Sanjay Ranka, My T. Thai, Takashi Washio, and Xindong Wu, editors, *IEEE International Conference on Data Mining, ICDM 2022, Orlando, FL, USA, November 28 - Dec. 1, 2022*, pages 1287–1292. IEEE, 2022. doi: 10.1109/ICDM54844.2022.00169. 1, 3

[10] Andreas Roth and Thomas Liebig. Rank collapse causes over-smoothing and over-correlation in graph neural networks. In Soledad Villar and Benjamin Chamberlain, editors, *Learning on Graphs Conference, 27-30 November 2023, Virtual Event*, volume 231 of *Proceedings of Machine Learning Research*, page 35. PMLR, 2023. 1, 2, 3, 6

[11] Andreas Roth. Simplifying the theory on over-smoothing. *CoRR*, abs/2407.11876, 2024. doi: 10.48550/ARXIV.2407.11876. 1, 2, 3, 8, 20

[12] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1725–1735. PMLR, 2020. 2, 3

[13] Michael Scholkemper, Xinyi Wu, Ali Jadbabaie, and Michael T. Schaub. Residual connections and normalization can provably prevent oversmoothing in gnns. *CoRR*, abs/2406.02997, 2024. doi: 10.48550/ARXIV.2406.02997. 2, 3

[14] Andreas Roth and Thomas Liebig. Transforming pagerank into an infinite-depth graph neural network. In Massih-Reza Amini, Stéphane Canu, Asja Fischer, Tias Guns, Petra Kralj Novak, and Grigorios Tsoumakas, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2022, Grenoble, France, September 19-23, 2022, Proceedings, Part II*, volume 13714 of *Lecture Notes in Computer Science*, pages 469–484. Springer, 2022. doi: 10.1007/978-3-031-26390-3_27. 2, 3

[15] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. Gated graph sequence neural networks. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. 2

[16] T. Konstantin Rusch, Benjamin Paul Chamberlain, Michael W. Mahoney, Michael M. Bronstein, and Siddhartha Mishra. 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*. OpenReview.net, 2023. 2, 3, 23

[17] Kenta Oono and Taiji Suzuki. 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*. OpenReview.net, 2020. 2, 6

[18] Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *CoRR*, abs/2006.13318, 2020. 6, 20

[19] Francesco Di Giovanni, James Rowbottom, Benjamin Paul Chamberlain, Thomas Markovich, and Michael M. Bronstein. Understanding convolution on graphs via energies. *Trans. Mach. Learn. Res.*, 2023, 2023. 2, 3

[20] Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *CoRR*, abs/1711.07553, 2017. 3, 18

[21] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Personalized embedding propagation: Combining neural networks on graphs with personalized pagerank. *CoRR*, abs/1810.05997, 2018. 3

[22] Ben Finkelshtein, Xingyue Huang, Michael M. Bronstein, and İsmail İlkan Ceylan. Cooperative graph neural networks. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. 3

[23] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 3

[24] Guohao Li, Chenxin Xiong, Guocheng Qian, Ali K. Thabet, and Bernard Ghanem. Deepergcn: Training deeper gcns with generalized aggregation functions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(11):13024–13034, 2023. doi: 10.1109/TPAMI.2023.3306930. 3

[25] Kaixiong Zhou, Xiao Huang, Daochen Zha, Rui Chen, Li Li, Soo-Hyun Choi, and Xia Hu. Dirichlet energy constrained learning for deep graph neural networks. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 21834–21846, 2021. 3

[26] Moshe Eliasof, Lars Ruthotto, and Eran Treister. Improving graph neural networks with learnable propagation operators. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 9224–9245. PMLR, 2023. 3

[27] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 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*, pages 3950–3957. AAAI Press, 2021. doi: 10.1609/AAAI.V35I5.16514. 3

[28] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. Principal neighbourhood aggregation for graph nets. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 3

[29] Shyam A. Tailor, Felix L. Opolka, Pietro Liò, and Nicholas Donald Lane. Do we need anisotropic graph neural networks? In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. 3

[30] Wei Jin, Xiaorui Liu, Yao Ma, Charu C. Aggarwal, and Jiliang Tang. Feature overcorrelation in deep graph neural networks: A new perspective. In Aidong Zhang and Huzefa Rangwala, editors, *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 709–719. ACM, 2022. doi: 10.1145/3534678.3539445. 3

[31] Uri Alon and Eran Yahav. 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*. OpenReview.net, 2021. 3

[32] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. 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*. OpenReview.net, 2022. 3

[33] Ralph Abboud, Radoslav Dimitrov, and İsmail İlkan Ceylan. Shortest path networks for graph property prediction. In Bastian Rieck and Razvan Pascanu, editors, *Learning on Graphs Conference, LoG 2022, 9-12 December 2022, Virtual Event*, volume 198 of *Proceedings of Machine Learning Research*, page 5. PMLR, 2022.

[34] Federico Barbero, Ameya Velingker, Amin Saberi, Michael M. Bronstein, and Francesco Di Giovanni. Locality-aware graph rewiring in gnns. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. 3

[35] Khang Nguyen, Nong Minh Hieu, Vinh Duc Nguyen, Nhat Ho, Stanley J. Osher, and Tan Minh Nguyen. Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 25956–25979. PMLR, 2023. 3

[36] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam, editors, *The Semantic Web - 15th International Conference, ESWC*

2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer, 2018. doi: 10.1007/978-3-319-93417-4_38. 3

[37] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. Composition-based multi-relational graph convolutional networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 3

[38] Susheel Suresh, Vinith Budde, Jennifer Neville, Pan Li, and Jianzhu Ma. Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns. In Feida Zhu, Beng Chin Ooi, and Chunyan Miao, editors, *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 1541–1551. ACM, 2021. doi: 10.1145/3447548.3467373. 3

[39] Yiding Yang, Zunlei Feng, Mingli Song, and Xinchao Wang. Factorizable graph convolutional networks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 3

[40] Jingwei Guo, Kaizhu Huang, Xinping Yi, and Rui Zhang. ES-GNN: generalizing graph neural networks beyond homophily with edge splitting. *CoRR*, abs/2205.13700, 2022. doi: 10.48550/ARXIV.2205.13700. 3

[41] Valentina Giunchiglia, Chirag Varun Shukla, Guadalupe Gonzalez, and Chirag Agarwal. Towards training gnns using explanation directed message passing. In Bastian Rieck and Razvan Pascanu, editors, *Learning on Graphs Conference, LoG 2022, 9-12 December 2022, Virtual Event*, volume 198 of *Proceedings of Machine Learning Research*, page 28. PMLR, 2022. 3

[42] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *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*, 2022. 3, 23

[43] Moshe Eliasof, Eldad Haber, and Eran Treister. ADR-GNN: advection-diffusion-reaction graph neural networks. *CoRR*, abs/2307.16092, 2023. doi: 10.48550/ARXIV.2307.16092. 3

[44] Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael M. Bronstein. Edge directionality improves learning on heterophilic graphs. In Soledad Villar and Benjamin Chamberlain, editors, *Learning on Graphs Conference, 27-30 November 2023, Virtual Event*, volume 231 of *Proceedings of Machine Learning Research*, page 25. PMLR, 2023. 3, 9, 22, 23

[45] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. 3

[46] Beatrice Bevilacqua, Moshe Eliasof, Eli A. Meirom, Bruno Ribeiro, and Haggai Maron. Efficient subgraph gnns by learning effective selection policies. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. 3

[47] Thomas N. Kipf and Max Welling. 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*. OpenReview.net, 2017. 4

[48] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In Carlotta Demeniconi and Ian Davidson, editors, *Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, Virtual Event, April 29 - May 1, 2021*, pages 333–341. SIAM, 2021. doi: 10.1137/1.9781611976700.38. 5

[49] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. 6, 8, 17

[50] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 1548–1554. ijcai.org, 2021. doi: 10.24963/IJCAI.2021/214. 6

[51] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. 6

[52] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. 6

[53] Alexander Frank Wells. *Structural inorganic chemistry*. Oxford University Press, USA, 2012. 6

[54] Teague Sterling and John J. Irwin. ZINC 15 - ligand discovery for everyone. *J. Chem. Inf. Model.*, 55(11):2324–2337, 2015. doi: 10.1021/ACS.JCIM.5B00559. 7, 18

[55] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *J. Mach. Learn. Res.*, 24:43:1–43:48, 2023. 7, 8

[56] Vijay Prakash Dwivedi, Ladislav Rampásek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *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*, 2022. 7, 18

[57] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 7, 18

[58] Jan Tönshoff, Martin Ritzert, Eran Rosenbluth, and Martin Grohe. Where did the gap go? reassessing the long-range graph benchmark. *Trans. Mach. Learn. Res.*, 2024, 2024. 7, 18, 20, 22

[59] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. 8

[60] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5449–5458. PMLR, 2018. 8

[61] Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 21618–21629, 2021. 8

[62] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 9, 22

[63] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser-Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 20887–20902, 2021. 9, 23

[64] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. 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*. OpenReview.net, 2023. 9

[65] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1024–1034, 2017. 17

[66] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. 18, 20, 21

[67] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *CoRR*, abs/1903.02428, 2019. URL http://arxiv.org/abs/1903.02428. 18

[68] Sandeep Singh, Kumardeep Chaudhary, Sandeep Kumar Dhanda, Sherry Bhalla, Salman Sadullah Usmani, Ankur Gautam, Abhishek Tuknait, Piyush Agrawal, Deepika Mathur, and Gajendra P. S. Raghava. Satpdb: a database of structurally annotated therapeutic peptides. *Nucleic Acids Res.*, 44(Database-Issue):1119–1126, 2016. doi: 10.1093/NAR/GKV1114. 18

[69] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *J. Complex Networks*, 9(2), 2021. doi: 10.1093/COMNET/CNAB014. 22

[70] Bronwyn H Hall, Adam B Jaffe, and Manuel Trajtenberg. The nber patent citation data file: Lessons, insights and methodological tools, 2001. 23

[71] Zekun Tong, Yuxuan Liang, Changsheng Sun, Xinke Li, David S. Rosenblum, and Andrew Lim. Digraph inception convolutional networks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 23

[72] Xitong Zhang, Yixuan He, Nathan Brugnone, Michael Perlmutter, and Matthew J. Hirn. Magnet: A neural network for directed graphs. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 27003–27015, 2021. 23

[73] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 23

[74] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 23

[75] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Improving graph neural networks with simple architecture design. *CoRR*, abs/2105.07634, 2021. URL https://arxiv.org/abs/2105.07634. 23

[76] Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 13242–13256. PMLR, 2022. 23

[77] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 40–48. JMLR.org, 2016. 23

[78] Nicolas Keriven. Not too little, not too much: a theoretical analysis of graph (over)smoothing. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors,

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, 2022. 24

A Mathematical Details

A.1 Proof of Theorem 4.4

Proof. We rewrite the rank-one matrix $\mathbf{X} = \mathbf{u}\mathbf{v}^T$ for some $\mathbf{u} \in \mathbb{R}^n$ and $\mathbb{R}^d$. Each $\mathbf{A}_p\mathbf{X}\mathbf{W}_p = \mathbf{u}_p\mathbf{v}_p^T$ is a rank-one matrix with $\mathbf{u}_p = \mathbf{A}_p\mathbf{u}$ and $\mathbf{v}_p^T = \mathbf{v}^T\mathbf{W}_p$. All $\mathbf{v}_p$ are linearly independent to the set of other $\{\mathbf{v}_1, \dots, \mathbf{v}_l\} \setminus \mathbf{v}_p$, as this holds for almost every $\mathbf{W}_1, \dots, \mathbf{W}_l$ with respect to the Lebesgue measure.

We rewrite $\mathbf{x}'_i$ and $\mathbf{x}'_j$ as $\mathbf{x}'_i = \sum_{p=1}^{l} \mathbf{u}_p[i]\mathbf{v}_p^T$. We further express this as a vector-matrix product $\mathbf{x}'_i = \mathbf{u}[i]^T\mathbf{V}$ where $\mathbf{u}[i] = [\mathbf{u}_1[i] \quad \dots \quad \mathbf{u}_l[i]]$ and $\mathbf{V}$ contains $\mathbf{v}_1, \dots, \mathbf{v}_l$ stacked as rows. We observe that $\mathbf{x}'_i$ and $\mathbf{x}'_j$ are linearly independent if and only if $\mathbf{u}[i]$ and $\mathbf{u}[j]$ are linearly independent.

For $\mathbf{u}[i]$ and $\mathbf{u}[j]$ to be linearly dependent, there must exist a scalar $c \in \mathbb{R}$ such that $\mathbf{u}[i] - c \cdot \mathbf{u}[j] = 0$. Thus, we also require element-wise for each $i = 1, \dots, l$ to hold that $\mathbf{A}_k[i] - m \cdot \mathbf{A}_k[j]\mathbf{u} = 0$. This is further true when $\mathbf{A}_k[i] - m\mathbf{A}_k[j] = 0$ for a.e. $\mathbf{u}$. By our assumption that nodes v_i and v_j are structurally independent, this cannot be satisfied for all elements simultaneously for any m. □

A.2 Proof of Theorem 4.5

Proof. Let $\mathbf{d}^{i_1}, \dots, \mathbf{d}^{i_l}$ with $i_k \in [n]$ be l linearly independent rows of $\mathbf{D}$. Each $\mathbf{x}'_{i_p}$ is linearly independent to the combination of all other $\mathbf{x}'_{i_1}, \dots, \mathbf{x}'_{i_{p-1}}, \dots, \mathbf{x}'_{i_{p+1}}, \dots, \mathbf{x}'_{i_l}$.

This Theorem then follows Theorem 4.4 by considering a given $\mathbf{x}'_{i_p}$ and replacing $\mathbf{u}_p[j]$ by any combination of the vectors from the other nodes. By injectivity of σ, representations remain linearly independent after applying σ for a.e. $\mathbf{W}_1, \dots, \mathbf{W}_l$. □

A.3 Proof of Corollary 5.1

Proof. For all graphs k and all nodes i, we have $d_k^i = 1$. Thus, all node pairs are structurally dependent. □

A.4 Proof of Proposition 5.3

Proof. Let i be a root node of $\mathcal{E}_1$ that is not a root node of $\mathcal{E}_2$. Thus, $s_1^i = 0$, while $s_2^i \neq 0$. For any non-root node j of $\mathbf{A}_1$, we have $s_1^j \neq 0$, ensuring structural independence of i and j for any s_2^j. □

B Experimental Details

In this section, we provide additional details regarding our experiments. All experiments were run on an internal cluster and separately on H100 GPUs, each on a single H100 GPU and an Intel Xeon 8468 Sapphire CPU.

B.1 MRS-MPNNs

MRS-SAGE. We define the MRS-SAGE version of the SAGE convolution [65] as

$$\left[\text{MRS-SAGE}\left(\mathbf{X}, \mathcal{E}, f\right)\right]_i = \mathbf{W}\mathbf{x}_i + \sum_{j \in N_i} \frac{1}{d_i} \mathbf{W}_{f(v_i, v_j)}\mathbf{x}_j \,, \tag{11}$$

where $\mathbf{W} \in \mathbb{R}^{d \times d'}$ is the additional feature transformation of the previous state $\mathbf{x}_i$.

MRS-GAT. For GAT [49], the corresponding MRS form is

$$\left[\text{MRS-GAT}(\mathbf{X}, \mathcal{E})\right]_i = \sum_{j \in N_i} \alpha_{ij} \mathbf{W}_{f(v_i, v_j)}\mathbf{x}_j \tag{12}$$

where $\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}_{f(v_i,v_j)} || \mathbf{W}_{f(v_i,v_j)}]))}{\sum_{k \in N_i} \exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}_{f(v_i,v_j)} || \mathbf{W}_{f(v_i,v_j)}]))}$ with $\mathbf{a} \in \mathbb{R}^{d'}$. In all experiments, GAT and MRS-GAT utilize two heads.

MRS-GIN. As the transformation in GIN is applied after the aggregation and combination with the previous state, we apply a different GIN instantiation on each edge set:

$$\text{MRS-GIN}_{\text{DA}}(\mathbf{X}, \mathcal{E}) = \text{GIN}(\mathbf{X}, \mathcal{E}_1) + \text{GIN}(\mathbf{X}, \mathcal{E}_2) + \text{GIN}(\mathbf{X}, \mathcal{E}_3)\,. \tag{13}$$

MRS-GatedGCN. We adapt GatedGCN [20] and its implementation given by Dwivedi et al. [66]:

$$[\text{MRS-GatedGCN}(\mathbf{X}, \mathcal{E})]_i = \mathbf{A}\mathbf{x}_i + \sum \frac{\sigma(\mathbf{D}\mathbf{x}_i + \mathbf{E}\mathbf{x}_j + \mathbf{C}\mathbf{e}_{ij}) \odot \mathbf{B}_{f(v_i,v_j)}\mathbf{x}_j}{\sum_{j \in N_i} \sigma(\mathbf{D}\mathbf{x}_i + \mathbf{E}\mathbf{x}_j + \mathbf{C}\mathbf{e}_{ij}) + \epsilon} \tag{14}$$

where $\mathbf{A}, \mathbf{C}, \mathbf{D}, \mathbf{E} \in \mathbb{R}^{d \times d'}$ and $\mathbf{B_1}, \ldots, \mathbf{B_l} \in \mathbb{R}^{d \times d}$ are linear transformations, $\mathbf{e}_{ij} \in \mathbb{R}^d$ are edge attributes, σ is the sigmoid function, and $\epsilon = 1e-6$ is a small constant.

B.2 Improving the Learning Process

Our implementation is built on the Long Range Graph Benchmark (LRGB) [56, 58] which is available under the MIT license. It is based on PyTorch Geometric [67]. We add our models without making changes to the optimization and data construction parts.

B.2.1 Models and Optimization

All models perform k iterations of message-passing, with ReLU as a non-linear activation function. We reuse the standard optimization process from [56] and use the AdamW [57] optimizer with a cosine learning rate schedule. The cross-entropy loss is used for optimization, and the average precision (AP) as a metric. All models use at most $500\,000$ parameters to ensure fairness.

B.2.2 Datasets

ZINC. We utilize the ZINC dataset [54] that contains $250\,000$ chemical compounds that are represented as graphs. We also utilize the ZINC12k subset that contains a subset of $12\,000$ chemical compounds. Each node represents an atom, and each edge is a bond between two atoms. On average, a graph has around 23 nodes and 50 edges. Node features are given as a single value indicating its corresponding type of heavy atom. We do not utilize edge features. The objective is given as a graph regression task, which corresponds to the prediction of the constrained solubility of the molecule. The mean absolute error (MAE) is used as the loss function and for the score. Each experiment on ZINC is repeated for three random seeds. ZINC is freely commercially available under the license DbCL.

Peptides-Func. The Peptides dataset consists of $15\,535$ peptides, which are short molecular chains [68]. As with ZINC, nodes of the graph represent atoms and edges the bonds between them. They are part of the LRGB as peptides have a large diameter while each node has a low average degree of around 2. Thus, it is argued that this dataset requires models that can combine distant information in the graph, i.e., models with many layers. Node and edge features are constructed using molecular SMILES based on the atom types. This dataset was released under license CC BY-NC 4.0.

The task is to predict the molecular properties of each peptide, i.e., a multi-label graph classification task. Each graph belongs, on average, to 1.65 of 10 classes. As proposed by Dwivedi et al. [56], the cross-entropy (CE) loss is used for optimization, and the unweighted mean average precision (AP) as the metric. We follow their same data split, i.e., 70% for training and 15% for validation and testing.

The resulting representations are globally aggregated using the mean and mapped to class probabilities using a linear layer.

Peptides-Struct. The dataset is the same as used for Peptides-Func. The task is to predict five continuous geometric properties of the peptides, i.e., a multi-label graph regression task. The same data split is utilized. The MAE is used for both optimization and as the target metric.

B.2.3 Orderings

For all orderings, we compute a single value r_i per node v_i and construct the strict partial ordering $i \prec j \Leftrightarrow r_i < r_j$.

Table 5: Best hyperparameters for results in Table 1

Dataset		ZINC	
Parameter	Split	LR	Layers
GCN	Train	0.001	8
	Val	0.0003	8
MRS-GCN$_{\text{DA}}$ (random)	Train	0.001	8
	Val	0.003	16
MRS-GCN$_{\text{DA}}$ (Features)	Train	0.001	8
	Val	0.001	8
MRS-GCN$_{\text{DA}}$ (PPR)	Train	0.001	8
	Val	0.001	8
MRS-GCN$_{\text{DA}}$ (Degree)	Train	0.0003	8
	Val	0.0003	8

Random. For the random ordering, we assign each node $v_i \in \mathcal{V}$ a unique random index $r_i \in [0, |\mathcal{V}|]$. This index is consistent across layers and epochs. The edges within each computational graph will be similar to those from different computational graphs. Thus, we expect the optimal solution to be achieved when all transformations are equal.

Features. This ordering utilizes the initial node features $\mathbf{x}_i \in \mathbb{R}^d$ by summing $r_i = \sum_{c \in [d]} \mathbf{x}_{ic}$ over all features of that node.

PPR. Here, we perform 15 iterations of Personalized PageRank (PPR) with a restart probability $\alpha = 0.1$. This provides a finer node centrality measure as opposed to the node degree. A finer ordering will have fewer edges in the third computational graph, but the similarity between edges in each computational graph may be lowered. A node's role in a graph is connected to its centrality, e.g., influential persons in social networks have many connections.

Degree. As a coarser node centrality measure, we consider the node degree $r_i = d_i$. For molecular graphs, the node degree is closely connected with its role within the molecule. The degree is also much more efficient to compute compared to PPR.

B.2.4 Comparing Partial Orderings (Table 1)

Experimental Setup. To compare the performances of the partial orderings we considered, we evaluate them on the ZINC task. All models have a fixed hidden dimension of 64. We tune the learning rate for all models with values $\in \{0.003, 0.001, 0.0003\}$ and the number of layers $\in \{1, 2, 4, 8, 16, 32\}$. All experiments are repeated for three random seeds. Best hyperparameters are presented in Table 5. We compute the ordering and the split once in each forward pass. The symmetric normalization of the adjacency matrix is performed once per forward pass for GCN and MRS-GCN. Displayed runtimes are for eight-layer models. These experiments run for a total of around 60 hours on one H100 GPU.

Runtime. We provide an additional in-depth runtime analysis in Table 6. We measure the time that applying the steps for performing normalization, ordering, splitting, applying feature transformations, and aggregating messages take. Normalization, ordering, and splitting are fixed throughout training, so they only have to be calculated once. As the same total number of parameters is used for all models, applying the three transformations $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$ only takes around 25% more time. Aggregating messages from all computational graphs takes an additional time of around 15%. In total, the differences are slightly lower, as all other computations are equivalent.

B.2.5 Preventing Rank Collapse (Figure 2)

We utilize random graphs of the ZINC dataset. We perform a single linear transformation to change the feature dimension to 16. We then apply message-passing iterations, each followed by a ReLU activation. Each iteration maintains the feature dimension as 16. Feature transformations are not

Table 6: Execution times in milliseconds for applying each step a single time on a batch of 32 graphs of the ZINC dataset. Normalization refers to applying symmetric degree normalization on the adjacency matrix, ordering refers to calculating a single scalar per node, and splitting refers to assigning each edge to one of the three edge sets based on the ordering. Transformation describes applying $\mathbf{W}$ or $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$ on the node state, and aggregation means collecting the weighted sum over all utilized edge relations.

METHOD	NORMALIZATION	ORDERING	SPLITTING	TRANSFORMATION	AGGREGATION
GCN	0.129	-	-	0.044	0.091
MRS-GCN$_{\text{DA}}$ (RANDOM)	0.130	0.019	0.249	0.051	0.097
MRS-GCN$_{\text{DA}}$ (FEATURES)	0.131	0.022	0.268	0.054	0.106
MRS-GCN$_{\text{DA}}$ (PPR)	0.128	1.150	0.212	0.055	0.092
MRS-GCN$_{\text{DA}}$ (DEGREE)	0.128	0.042	0.261	0.055	0.104

Table 7: Best hyperparameters for results in Table 2

DATASET	ZINC (TRAIN)		ZINC (TEST)	
PARAMETER	LR	LAYERS	LR	LAYERS
GCN	0.0003	16	0.0003	16
MRS-GCN$_{\text{DA}}$	0.0003	8	0.0001	8
SAGE	0.0003	8	0.0003	8
MRS-SAGE$_{\text{DA}}$	0.0003	8	0.0003	8
GAT	0.0003	8	0.0003	8
GAT+DEGREE	0.0003	8	0.0003	8
MRS-GAT$_{\text{DA}}$	0.0003	8	0.0003	8
GIN	0.0003	8	0.0003	8
MRS-GIN$_{\text{DA}}$	0.0003	8	0.0001	8
GATEDGCN	0.0003	8	0.0003	8
MRS-GATEDGCN$_{\text{DA}}$	0.0003	8	0.0003	8

shared between iterations. The Rank-one distance is defined as

$$\text{ROD}(\mathbf{X}) = \left\| \frac{\mathbf{X}}{\|\mathbf{X}\|} - \frac{\mathbf{u}\mathbf{v}^T}{\|\mathbf{u}\mathbf{v}^T\|} \right\| \tag{15}$$

where $\mathbf{u} \in \mathbb{R}^n$ is the column $\mathbf{v} \in \mathbb{R}^d$ the row of $\mathbf{X}$ with the largest norm [11]. All norms are nuclear norms. As this metric generalizes the Dirichlet energy [18], constructing models that keep ROD constant, also prevent over-smoothing. It is calculated after the ReLU activation. We repeat this experiment for 50 random seeds. These experiments run for less than one minute on a CPU.

B.2.6 Details on Table 2

For ZINC, we tune the number of layers in $\{1, 2, 4, 8, 16, 32\}$ and the base learning rate in $\{0.001, 0.0003, 0.0001\}$. All runs utilize a cosine learning rate schedule with a maximum of 500 epochs for ZINC. As given by our reference implementation [66], a batch size of 32 is used. The hidden dimension of each model is reduced until less than 500 000 parameters are used. As an example, the hidden dimensions of the eight-layer models are shown in Table 8. Best hyperparameters for each experiment are displayed in Table 7. These experiments require around 4500 hours on an H100 GPU.

For Peptides-Func, we tune the number of layers in $\{1, 2, 4, 8, 16, 32\}$ and the base learning rate in $\{0.005, 0.001, 0.0005\}$ using a grid search. As given by our reference implementation [58], a batch size of 200 is used. Best hyperparameters for each experiment are displayed in Table 10. These experiments require around 70 hours on an H100 GPU. Results are presented in Table 9.

B.2.7 Details on Figure 3

We track the training loss on the full ZINC dataset. Hyperparameters are selected based on the minimal achieved final training loss for each model. We repeated all settings for three random seeds.

Table 8: Hidden dimensions and total parameter count for eight-layer models on ZINC and ZINC12k.

DATASET	HIDDEN DIMENSION	PARAMETERS
GCN	247	498200
MRS-GCN$_{\text{DA}}$	143	496656
SAGE	175	497176
MRS-SAGE$_{\text{DA}}$	124	497117
GAT	174	497119
GAT+DEGREE	174	497151
MRS-GAT$_{\text{DA}}$	101	497022
GIN	143	498928
MRS-GIN$_{\text{DA}}$	101	497830
GATEDGCN	110	495551
MRS-GATEDGCN$_{\text{DA}}$	82	495363

Table 9: Mean MAE and standard deviations over three runs on the Peptides-Func dataset (pairwise best results marked in **bold**). The learning rate and the number of layers are tuned. Best hyperparameters are provided in brackets. Train MAE and AP are the overall minimum, while test AP is based on the best validation AP.

METHOD	PEPTIDES-FUNC			PEPTIDES-STRUCT	
	TRAIN (MAE)	TRAIN (AP)	TEST (AP)	TRAIN (MAE)	TEST (MAE)
GCN	0.033 ± 0.023	0.989 ± 0.004	0.574 ± 0.005	0.146 ± 0.032	0.303 ± 0.009
MRS-GCN$_{\text{DA}}$	$\mathbf{0.003} \pm 0.000$	$\mathbf{0.999} \pm 0.000$	$\mathbf{0.588} \pm 0.011$	$\mathbf{0.052} \pm 0.004$	$\mathbf{0.301} \pm 0.007$
SAGE	0.011 ± 0.004	$\mathbf{0.999} \pm 0.000$	0.566 ± 0.011	0.168 ± 0.004	0.317 ± 0.0001
MRS-SAGE$_{\text{DA}}$	$\mathbf{0.002} \pm 0.000$	$\mathbf{0.999} \pm 0.000$	$\mathbf{0.615} \pm 0.008$	$\mathbf{0.055} \pm 0.004$	$\mathbf{0.293} \pm 0.000$
GIN	0.031 ± 0.021	0.980 ± 0.021	0.553 ± 0.016	0.228 ± 0.012	0.307 ± 0.001
MRS-GIN$_{\text{DA}}$	$\mathbf{0.007} \pm 0.002$	$\mathbf{0.999} \pm 0.000$	$\mathbf{0.576} \pm 0.007$	$\mathbf{0.149} \pm 0.013$	$\mathbf{0.301} \pm 0.002$

All runs utilize a cosine learning rate schedule with a maximum of 500 epochs. As given by Dwivedi et al. [66], a batch size of 32 is used. Including hyperparameter optimization, these experiments took 2700 hours on an H100 GPU.

B.2.8 Details on Table 3

Jumping Knowledge. We store the output of each layer after applying the non-linearity into a set $\mathcal{X} = \{\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(k)}\}$. The resulting representation is then obtained by either taking the element-wise maximum, or concatenating all representation per node. This output is used instead of the output of the last message-passing layer.

Table 10: Best hyperparameters for the results in Table 9. Entries are formatted as optimal learning rate / number of layers.

METHOD	PEPTIDES-FUNC			PEPTIDES-STRUCT	
	TRAIN (MAE)	TRAIN (AP)	TEST (AP)	TRAIN (MAE)	TEST (MAE)
GCN	0.005 / 4	0.005 / 4	0.001 / 8	0.005 / 8	0.0005 / 32
MRS-GCN$_{\text{DA}}$	0.005 / 16	0.005 / 16	0.001 / 16	0.005 / 8	0.001 / 32
SAGE	0.001 / 16	0.001 / 16	0.001 / 8	0.005 / 8	0.0005 / 16
MRS-SAGE$_{\text{DA}}$	0.001 / 16	0.001 / 16	0.001 / 16	0.005 / 8	0.0005 / 32
GIN	0.0001 / 8	0.0001 / 8	0.001 / 8	0.001 / 8	0.0005 / 16
MRS-GIN$_{\text{DA}}$	0.001 / 8	0.001 / 8	0.0005 / 16	0.001 / 8	0.0005 / 32

Table 11: Standard deviations for the results in Table 3

Method	1	2	4	8	16	32
GCN	0.002	0.001	0.004	0.011	0.014	0.002
MRS-GCN$_{DA}$	0.002	0.007	0.011	0.003	0.014	0.006
GCN + Res	0.003	0.004	0.002	0.014	0.021	0.019
MRS-GCN$_{DA}$ + Res	0.001	0.001	0.011	0.002	0.003	0.004
GCN + JK	0.004	0.002	0.003	0.011	0.009	0.010
MRS-GCN$_{DA}$ + JK	0.004	0.000	0.003	0.003	0.005	0.009
GCN + LapPE	0.017	0.008	0.005	0.018	0.031	0.013
MRS-GCN$_{DA}$ + LapPE	0.020	0.060	0.014	0.020	0.009	0.002

Table 12: Best hyperparameters for results in Table 3

METHOD	1	2	4	8	16	32
GCN	0.0003	0.0003	0.0003	0.0003	0.0003	0.001
MRS-GCN$_{DA}$	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003
GCN + RES	0.0003	0.0003	0.0003	0.001	0.001	0.001
MRS-GCN$_{DA}$ + RES	0.0003	0.0003	0.0003	0.001	0.001	0.003
GCN + JK	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003
MRS-GCN$_{DA}$ + JK	0.0003	0.0003	0.001	0.0003	0.0003	0.0003
GCN + LAPPE	0.001	0.0003	0.0003	0.0003	0.001	0.0003
MRS-GCN$_{DA}$ + LAPPE	0.001	0.003	0.001	0.0003	0.0003	0.0003

Residual Connections. This technique adds the previous state to the message-passing operation's output for each layer after the non-linearity.

Laplacian Positional Encoding. We obtain Laplacian positional encodings based on eight frequencies and concatenate these features to the initial node representations.

Optimization. For each model type and layer count, we tune the learning rate in $\{0.001, 0.0003, 0.0001\}$. Reported test scores are based on the epoch of the best validation score. All runs utilize a cosine learning rate schedule with a maximum of 400 epochs. As given by Tönshoff et al. [58], a batch size of 32 is used. These experiments require a total of around 110 hours on an H100 GPU.

B.3 Comparison with State-of-the-Art

Our models are integrated into the implementation of Dir-GNNs [44]. This implementation is available under the MIT license.

B.4 Memory Costs

We present the empirical memory costs in Table 13. As the GCN and the MRS-GCN$_{DA}$ utilize the same number of parameters, their memory consumption is almost identical.

B.4.1 Datasets

Chameleon and Squirrel. These two datasets are based on pages about particular topics in Wikipedia. Nodes represent articles on that topic and edges their links. Node features are constructed as the appearance of particular nouns [69]. The task is to classify each article based on their average monthly traffic [62]. Chameleon consists of 2277 nodes and 36 101 edges. Squirrel consists of 5201 nodes and 217 073 edges. To the best of our knowledge, the dataset was released without a license.

Table 13: Memory usage during training of ZINC12k. Values are averaged over five runs.

METHOD	GPU MEMORY	MAIN MEMORY
GCN	1.918 GB	1.218 GB
MRS-GCN$_{\text{DA}}$	1.914 GB	1.265 GB

Table 14: Best hyperparameters for the results in Table 4.

DATASET	LEARNING RATE	LAYERS	DROPOUT RATIO
CHAMELEON	0.005	6	0.2
SQUIRREL	0.001	6	0.0
ROMAN-EMPIRE	0.005	6	0.2
ARXIV-YEAR	0.005	5	0.6
SNAP-PATENTS	0.01	6	0.0

Arxiv-Year. In this dataset, nodes correspond to publications, and an edge is constructed when a publication cites another. The task is to classify the publication year into one of five time spans. It consists of 169 343 nodes and 1 166 243 edges. Nodes are given by word 128-dimensional embeddings of the title and abstract of the corresponding publication. Arxiv-Year is released under the ODC-BY license.

Snap-Patents. Nodes correspond to patents, for which the year it was granted should be classified into one of five time spans. Edges similarly correspond to citations between patents. This dataset consists of 2 923 922 nodes and 13 975 791 edges. This dataset was released without a license by Hall et al. [70].

B.4.2 Implementational Details

We compare to several state-of-the-art methods for directed graphs, namely DiGCN [71], MagNet [72], and Dir-GNN [44], and state-of-the-art methods for heterophilic graphs, namely H_2GCN [73], GPR-GNN [74], LINKX [63], FSGNN [75], ACM-GCN [42], GloGNN [76], and Gradient Gating [16].

We use the same model as Rossi et al. [44], with the only change being the exchange of each MPNN module with the corresponding MRS-MPNN$_{\text{DA}}$. The models consist of k layers of message-passing, each followed by ReLU and potentially Dropout. All representations are normalized by the $L2$-norm $\|\mathbf{X}\|_2$. Final representations are obtained by Knowledge Knowledge, either concatenating all intermediate states (cat) or taking the element-wise maximum (max). Node degree is used as ordering for all experiments. We tune MRS-Dir-GNN$_{\text{DA}}$ using the same hyperparameters and their ranges as performed for Dir-GNN using the same implementation: The learning rate $\in \{0.01, 0.005, 0.001, 0.0005\}$, number of layers $\in \{4, 5, 6\}$, jumping knowledge $\in \{\text{cat}, \text{max}\}$, dropout $\in \{0.0, 0.2, 0.4, 0.6\}$. We use their optimal values for hidden feature dimension $\in \{32, 64, 128, 256, 512\}$, normalization $\in \{\text{True}, \text{False}\}$ and $\alpha \in \{0., 0.5, 1\}$ for each task. As given in their implementation, the patience of stopping training based on not improving the validation accuracy is set to 200 for Roman-Empire, Snap-Patents, and Arxiv-Year and to 400 for Squirrel and Chameleon. Consequently, MRS-SAGE$_{\text{DA}}$ is used for Roman-Empire and MRS-GCN$_{\text{DA}}$ for the other datasets. The best-performing hyperparameters are presented in Table 14.

B.4.3 Runtime

On an H100 GPU, each run for Chameleon takes around 10 seconds, for Squirrel 20 seconds, for Roman-Empire 90 seconds, for Arxiv-Year 15 minutes, and for Snap-Patents 30 minutes. In total, these Experiments take around 150 GPU hours. We estimate the time of our preliminary experiments with various versions of MRS-MPNNs to additional 1500 hours.

B.5 Additional Experiments on Homophilic Datasets

We additionally conducted experiments on three datasets for homophilic node classification, namely Cora, CiteSeer, and PubMed [77]. We evaluate the GCN and the MRS-GCN$_{\text{DA}}$. We tune the learning

Table 15: Test accuracies for homophilic datasets.

METHOD	CORA (ACC)	CITESEER (ACC)	PUBMED (ACC)
GCN	81.1 ± 0.9	66.6 ± 1.0	75.5 ± 1.2
MRS-GCN$_{\text{DA}}$	80.0 ± 1.2	62.8 ± 1.7	76.2 ± 0.7

Table 16: Best hyperparameters for Table 15. Entries are formatted as Learning rate / dropout ratio / number of layers.

METHOD	CORA (ACC)	CITESEER (ACC)	PUBMED (ACC)
GCN	$0.01/0.0/2$	$0.03/0.6/2$	$0.01/0.6/4$
MRS-GCN$_{\text{DA}}$	$0.03/0.6/2$	$0.03/0.4/2$	$0.01/0.4/4$

rate in $\{0.03, 0.01, 0.003\}$, the dropout ratio in $\{0.0, 0.2, 0.4, 0.6, 0.8\}$, and the number of layers in $\{1, 2, 4, 8, 16, 32\}$ for both methods. These experiments take around three hours on a single H100 GPU.

Results are presented in Table 15. The accuracy achieved by MRS-GCN$_{\text{DA}}$ is lower for Cora and CiteSeer, which indicates that splitting edges based on the degree does not hold relevant information for this task. Instead, these homophilic tasks are known to benefit from smoothing dynamics [78]. Having $\mathbf{W}_1 = \mathbf{W}_2 = \mathbf{W}_3$ would be better, but this solution is apparently harder to find. Different edge-splitting functions would likely result in a higher accuracy, e.g., by splitting based on inter- and intra-community edges.