BREAKING THE EXPRESSIVE BOTTLENECKS OF GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recently, the Weisfeiler-Lehman (WL) graph isomorphism test was used to measure the expressiveness of graph neural networks (GNNs), showing that the neighborhood aggregation GNNs were at most as powerful as 1-WL test in distinguishing graph structures. There were also improvements proposed in analogy to k-WL test ($k > 1$). However, the aggregators in these GNNs are far from injective as required by the WL test, and suffer from weak distinguishing strength, making it become expressive bottlenecks. In this paper, we improve the expressiveness by exploring powerful aggregators. We reformulate aggregation with the corresponding aggregation coefficient matrix, and then systematically analyze the requirements of the aggregation coefficient matrix for building more powerful aggregators and even injective aggregators. It can also be viewed as the strategy for preserving the rank of hidden features, and implies that basic aggregators correspond to a special case of low-rank transformations. We also show the necessity of applying nonlinear units ahead of aggregation, which is different from most aggregation-based GNNs. Based on our theoretical analysis, we develop two GNN layers, ExpandingConv and CombConv. Experimental results show that our models significantly boost performance, especially for large and densely connected graphs.

1 INTRODUCTION

Graphs are ubiquitous in the real world. Social networks, traffic networks, knowledge graphs, and molecular structures are typical graph-structured data. Graph Neural Networks (GNNs) Scarselli et al. (2008); Gori et al. (2005), leveraging the power of neural networks to graph-structured data, have a rapid development recently Kipf & Welling (2016); Hamilton et al. (2017); Bronstein et al. (2017); Gilmer et al. (2017); Duvenaud et al. (2015).

Expressive power of GNNs measures their abilities to represent different graph structures Sato (2020). It decides the performance of GNNs where the awareness of graph structures is required, especially on large graphs with complex topologies. The neighborhood aggregation scheme (or message passing) follows the same pattern with weisfiler-lehman (WL) graph isomorphism test Weisfeiler & Leman (1968) to encode graph structures, where node representations are computed iteratively by aggregating transformed representations of its neighbors with structural information learned implicitly. Therefore, the WL test is used to measure the expressiveness of GNNs. Unfortunately, general GNNs are at most as powerful as 1-order WL test Morris et al. (2019); Xu et al. (2019). There is also work trying to improve the expressiveness that are beyond 1-order WL test Maron et al. (2019); Morris et al. (2019); Chen et al. (2019); Li et al. (2020b); Vignac et al. (2020). However, the weak distinguishing strength of aggregators is the fundamental limitation. The expressiveness analysis measured by the WL test assumes that aggregators are injective, which is usually unreachable. Therefore, this motivates us to investigate the following questions: What are the key factors to limit the expressiveness of GNN? and how to break these limitations?

Aggregators are permutation invariant functions that operate on sets while preserving permutation invariance. Zaheer et al. (2017) first theoretically studied permutation invariant functions and provided a family of functions to which any permutation invariant function must belong. Xu et al. (2019) extended it on multisets but only for countable space. Corso et al. (2020) further extended it to uncountable space. Murphy et al. (2018) and Murphy et al. (2019) expressed a permutation invariant function by approximating an average over permutation-sensitive functions with tractability

strategies. Dehmamy et al. (2019) showed that a single propagation rule applied in general GNNs is rather restrictive in learning graph moments Lin & Skiena (1995). They and Corso et al. (2020) improved the distinguishing strength of aggregation by leveraging multiple basic aggregators (SUM, MEAN, NORMALIZED MEAN, MAX/MIN, and STD). This strategy showed its effectiveness on tasks taken from classical graph theory.

In contrast to existing studies towards aggregators in GNNs, we provide a new GNN formulation, where the aggregation is represented as the multiplication of the corresponding hidden feature matrix of neighbors and the aggregation coefficient matrix. This new formulation enables us to answer the following questions: (i) when a GNN will lose its expressive power; (ii) How to build aggregators with higher distinguishing strength, even injective aggregators. Based on our theoretical analysis, we propose two GNN layers: ExpandingConv and CombConv, and evaluate them on general graph classification and graph regression tasks. Our key contributions are summarized as follows:

- We formalize the distinguishing strength of aggregators as a partial order, and theoretically show that the choice of aggregators can be bottlenecks of expressiveness. We also propose to apply nonlinear units ahead of aggregations to break the distinguishing strength limitations of aggregators as well as to achieve an implicit sampling mechanism.
- We reformulate the neighborhood aggregation with the aggregation coefficient matrix and then provide a theoretical point of view on building powerful aggregators and even injective aggregators.
- We propose ExpandingConv and CombConv layers which achieve state-of-the-art performance on a variety of graph tasks. We also show that multi-head GAT is one of the ExpandingConv implementations, which brings a theoretical explanation for its effectiveness.

2 PRELIMINARIES

2.1 NOTATIONS

For a graph $G(V, E)$, we denote the set of edges, nodes and node feature vectors respectively by E_G, V_G and X_G. $\mathcal{N}(v)$ represents the set of neighbors of v including itself, i.e., $\mathcal{N}(v) = \{u \in V_G | (u, v) \in E_G\} \cup \{v\}$. We use $[n]$ to denote the set $\{1, 2, ..., n\}$. $\{\{...\}\}$ represents a multi-set, i.e., a set with possibly repeating elements. Π_n represents the set of all permutations of the integers 1 to n. $\boldsymbol{h}_\pi$, where $\pi \in \Pi_{|h|}$, is a reordering of the elements of a sequence $\boldsymbol{h}$ according to π. Given a matrix $\boldsymbol{X} \in \mathbb{R}^{a \times b}$, $\boldsymbol{X}^T$ represents the transpose of $\boldsymbol{X}$, and $\text{vec}(\boldsymbol{X}) \in \mathbb{R}^{ab \times 1}$ represents the column stack of $\boldsymbol{X}$.

2.2 GRAPH NEURAL NETWORKS

Most GNNs adopt the neighborhood aggregation scheme Gilmer et al. (2017) to learn the node representations, which utilizes both node features and graph structures. In the k-th layer, the representation of node v

$$\boldsymbol{h}_v^{(k)} = \text{Update}(\boldsymbol{h}_v^{(k-1)}, \text{Aggregate}(\{\{\boldsymbol{h}_u^{(k-1)} | u \in \mathcal{N}(v)\}\})). \tag{1}$$

Aggregators in GNNs. An aggregator is a permutation invariant function Zaheer et al. (2017) with bounded size inputs. It satisfies: (i) size insensitive: an aggregator can take an arbitrary but finite size of inputs; (ii) permutation invariant: an aggregator is invariant to the permutation of input. There are a limited number of basic aggregators such as SUM, MEAN, NORMALIZED MEAN, MAX/MIN, STD, etc. Most proposed GNNs apply one of these aggregators. Sum-of-power mapping Zaheer et al. (2017) and normalized moments Corso et al. (2020) can also be used as aggregators and they allow for a variable number of aggregators.

3 PROPOSED MODEL

In this section, we first formalize the distinguishing strength of aggregators as a partial order, and show why basic aggregators used in popular GNNs become bottlenecks of expressiveness. Then, we analyze the requirements for building powerful aggregators and even injective aggregators. Finally, we introduce two GNN layers based on our theoretical analysis.

3.1 DISTINGUISHING STRENGTH OF AGGREGATORS

To ensure generality, our analysis of aggregators is always considered in multisets and uncountable case, where the inputs are continuous and with possibly repeating elements.

Distinguishing strength. The distinguishing strength of aggregator $\boldsymbol{f}_{aggr1}$ is stronger than $\boldsymbol{f}_{aggr2}$, denoted by $\boldsymbol{f}_{aggr1} \succeq \boldsymbol{f}_{aggr2}$, if and only if for any two multisets $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ where the number of elements can be different, $\boldsymbol{f}_{aggr2}(\boldsymbol{x}_1) \neq \boldsymbol{f}_{aggr2}(\boldsymbol{x}_2) \Rightarrow \boldsymbol{f}_{aggr1}(\boldsymbol{x}_1) \neq \boldsymbol{f}_{aggr1}(\boldsymbol{x}_2)$. Meanwhile, if there exist $\boldsymbol{x}_1'$ and $\boldsymbol{x}_2'$ such that $\boldsymbol{f}_{aggr1}(\boldsymbol{x}_1') \neq \boldsymbol{f}_{aggr1}(\boldsymbol{x}_2')$ but $\boldsymbol{f}_{aggr2}(\boldsymbol{x}_1') = \boldsymbol{f}_{aggr2}(\boldsymbol{x}_2')$, $\boldsymbol{f}_{aggr1}$ is strictly stronger than $\boldsymbol{f}_{aggr2}$, denoted by $\boldsymbol{f}_{aggr1} \succ \boldsymbol{f}_{aggr2}$. If $\boldsymbol{f}_{aggr1} \succeq \boldsymbol{f}_{aggr2} \succeq \boldsymbol{f}_{aggr1}$, we say these two aggregators have the same distinguishing strength, denoted by $\boldsymbol{f}_{aggr1} = \boldsymbol{f}_{aggr2}$. If there does not exist $\boldsymbol{f}_{aggr1} \succeq \boldsymbol{f}_{aggr2}$ or $\boldsymbol{f}_{aggr2} \succeq \boldsymbol{f}_{aggr1}$, we say $\boldsymbol{f}_{aggr1}$ and $\boldsymbol{f}_{aggr2}$ are incomparable.

Distinguishing strength is a partial order, and all aggregators form a poset. In this poset, the aggregators with the greatest distinguishing strength should be injective. With the definition of distinguishing strength, we can compare any two aggregators. The distinguishing strength of widely used aggregators SUM, MEAN, MAX/MIN is incomparable. One can easily give two multisets of elements that are distinguished by one aggregator but are not distinguished by the others.

Equivariant aggregator. $\boldsymbol{f}_{\text{aggr}} : \{\{\mathbb{R}^d\}\} \to \mathbb{R}^d$ is an equivariant aggregator if and only if $\boldsymbol{f}_{\text{aggr}}(\{\{\boldsymbol{T} \cdot \boldsymbol{x}_1, \boldsymbol{T} \cdot \boldsymbol{x}_2, \cdots, \boldsymbol{T} \cdot \boldsymbol{x}_n\}\}) = \boldsymbol{T} \cdot \boldsymbol{f}_{\text{aggr}}(\{\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}\})$ for any $\boldsymbol{T} \in \mathbb{R}^{m \times d}$ and $\{\{\boldsymbol{x}_i \in \mathbb{R}^d | i \in [n]\}\}$.

Widely used SUM and MEAN are equivariant aggregators but MAX/MIN is not. We denote $\boldsymbol{f}_{aggr1} \otimes \boldsymbol{f}_{aggr2}$ a new aggregator by combing $\boldsymbol{f}_{aggr1}$ and $\boldsymbol{f}_{aggr2}$ with $\boldsymbol{f}_{aggr1} \otimes \boldsymbol{f}_{aggr2}(X) = [\boldsymbol{f}_{aggr1}(X) || \boldsymbol{f}_{aggr2}(X)]$, where $||$ denotes concatenation.

Lemma 1. *(i) For any continuous function g, we have $g \circ \boldsymbol{f}_{aggr} \preceq \boldsymbol{f}_{aggr}$, and when g is injective, $\boldsymbol{f}_{aggr} = g \circ \boldsymbol{f}_{aggr}$; (ii) $\boldsymbol{f}_{aggr1} \otimes \boldsymbol{f}_{aggr2} \succeq \boldsymbol{f}_{aggr1}$ or $\boldsymbol{f}_{aggr2}$, and if $\boldsymbol{f}_{aggr1}$ and $\boldsymbol{f}_{aggr2}$ are incomparable, $\boldsymbol{f}_{aggr1} \otimes \boldsymbol{f}_{aggr2} \succ \boldsymbol{f}_{aggr1}$ or f_{aggr2}; (iii) If $\boldsymbol{f}_{aggr}$ is an equivariant aggregator, then $\boldsymbol{f}_{aggr}(\boldsymbol{T} \cdot \boldsymbol{x}_1, \boldsymbol{T} \cdot \boldsymbol{x}_2, \cdots, \boldsymbol{T} \cdot \boldsymbol{x}_n) \preceq \boldsymbol{f}_{aggr}(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)$ for any $\boldsymbol{T} \in \mathbb{R}^{m \times d}$ and $\{\{\boldsymbol{x}_i \in \mathbb{R}^d | i \in [n]\}\}$.*

We prove Lemma 1 in Appendix B. Lemma 1 indicates that aggregators become bottlenecks of distinguishing strength. For the equivariant aggregator, any linear transformation before aggregating and any transformation after aggregating have no contribution to the distinguishing strength. For SUM and MEAN, we have $g(\text{SUM}(\boldsymbol{T} \cdot \boldsymbol{x}_1, \boldsymbol{T} \cdot \boldsymbol{x}_2, \cdots, \boldsymbol{T} \cdot \boldsymbol{x}_n)) \preceq \text{SUM}(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)$ and $g(\text{MEAN}(\boldsymbol{T} \cdot \boldsymbol{x}_1, \boldsymbol{T} \cdot \boldsymbol{x}_2, \cdots, \boldsymbol{T} \cdot \boldsymbol{x}_n)) \preceq \text{MEAN}(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)$, where $\boldsymbol{T} \in \mathbb{R}^{m \times d}$, and g can be any continuous function. Based on Lemma 1, we can now compare the distinguishing strength of aggregation in some popular GNNs. GIN-0 sums all hidden features of neighbors at first, and then pass them to a 2-layer MLP. Therefore, when considering in a continuous input features space, the distinguishing strength of GIN-0 is at most as powerful as the SUM aggregator. GCN uses a NORMALIZED MEAN (denoted by nMEAN) aggregator. Given a node v and its neighbors, $n\text{MEAN}(v, u_1, \cdots, u_n) = \frac{1}{\sqrt{|\mathcal{N}(v)|}} \cdot (\frac{\boldsymbol{h}_v}{\sqrt{|\mathcal{N}(v)|}} + \frac{\boldsymbol{h}_{u_1}}{\sqrt{|\mathcal{N}(u_1)|}} + \cdots + \frac{\boldsymbol{h}_{u_n}}{\sqrt{|\mathcal{N}(u_n)|}})$. nMEAN is also an equivariant aggregator, and the distinguishing strength of aggregation in GCN is at most as powerful as nMEAN. GAT corresponds to the weighted SUM aggregation, where the weight coefficients are the functions of hidden features. This makes the distinguishing strength of GAT and SUM incomparable. Based on these observations, a potential approach to breaking the distinguishing strength limitation is to apply a nonlinear processing on inputs before aggregation.

3.2 BUILDING POWERFUL AGGREGATORS

In this section, we analyze the requirements for building more powerful aggregators and further injective aggregators. We first introduce a new representation of GNN layers which unifies several popular GNN layers. Given a node v and its neighbors $\mathcal{N}(v)$, our new formulation represents the GNN operation as follows:

$$\begin{cases} \boldsymbol{m}_v = \boldsymbol{f}_{\text{local}}(v) & \text{/* aggregation coefficients generation */} \\ \boldsymbol{r}_v^{(t)} = \boldsymbol{m}_{v\pi} \bar{\boldsymbol{h}}_{v\pi}^{(t-1)T} & \text{/* neighborhood aggregation */} \\ \boldsymbol{h}_v^{(t)} = \boldsymbol{f}_{\text{NN}}(\boldsymbol{r}_v^{(t)}) & \text{/* feature/structure extraction */.} \end{cases} \tag{2}$$

Here, $\boldsymbol{m}_v \in \mathbb{R}^{|\mathcal{N}(v)|}$ is the aggregation coefficients vector of node v. Note that $\boldsymbol{m}_v$ should be the mapping of local structures such as local topologies, node or edge features to ensure the same encoding of isomorphic graphs. $\bar{\boldsymbol{h}}_{v\pi}^{(t-1)T} = (\boldsymbol{h}_v^{(t-1)}, \boldsymbol{h}_{u_1}^{(t-1)}, \cdots, \boldsymbol{h}_{u_{|\mathcal{N}(v)|}}^{(t-1)})^T \in \mathbb{R}^{|\mathcal{N}(v)| \times d}$ is the matrix representation of v's neighbors according to a permutation π. $\boldsymbol{f}_{\text{NN}} : \mathbb{R}^d \to \mathbb{R}^d$ is a neural network that updates hidden feature $\boldsymbol{h}_v^{(t)}$ of node v with the aggregation result $\boldsymbol{r}_v^{(t)}$. We give this three-stage representation of GCN, GIN and GAT respectively as follows:

$$\text{GCN} : \begin{cases} \boldsymbol{m}_v = \frac{1}{\sqrt{|\mathcal{N}(v)|}}(\frac{1}{\sqrt{|\mathcal{N}(u_1)|}}, \frac{1}{\sqrt{|\mathcal{N}(u_2)|}}, \cdots, \frac{1}{\sqrt{|\mathcal{N}(u_{|\mathcal{N}(v)|})|}}) \\ \boldsymbol{r}_v^{(t)} = \boldsymbol{m}_{v\pi} \bar{\boldsymbol{h}}_{v\pi}^{(t-1)T} \\ \boldsymbol{h}_v^{(t)} = \sigma(\boldsymbol{W}\boldsymbol{r}_v^{(t)} + \boldsymbol{b}), \end{cases} \qquad \text{GIN0} : \begin{cases} \boldsymbol{m}_v = \boldsymbol{1}^{1 \times |\mathcal{N}(v)|} \\ \boldsymbol{r}_v^{(t)} = \boldsymbol{m}_{v\pi} \bar{\boldsymbol{h}}_{v\pi}^{(t-1)T} \\ \boldsymbol{h}_v^{(t)} = \text{MLP}(\boldsymbol{r}_v^{(t)}), \end{cases}$$

$$\text{GAT} : \begin{cases} \boldsymbol{m}_v^{(t)} = (\text{att}(\boldsymbol{h}_v^{(t-1)}, \boldsymbol{h}_{u_1}^{(t-1)}), \text{att}(\boldsymbol{h}_v^{(t-1)}, \boldsymbol{h}_{u_2}^{(t-1)}), \cdots, \text{att}(\boldsymbol{h}_v^{(t-1)}, \boldsymbol{h}_{u_{|\mathcal{N}(v)|}}^{(t-1)})) \\ \boldsymbol{r}_v^{(t)} = \boldsymbol{m}_{v\pi}^{(t)} \bar{\boldsymbol{h}}_{v\pi}^{(t-1)T} \\ \boldsymbol{h}_v^{(t)} = \sigma(\boldsymbol{W}\boldsymbol{r}_v^{(t)} + \boldsymbol{b}). \end{cases}$$

Their default formulations are given in Appendix A. In GCN, $\boldsymbol{m}_v$ is the function of node degrees. In GAT, $\boldsymbol{m}_v$ is the function of hidden features. Both of which ensure the same encoding of isomorphic graphs. From the above representations, the only difference between GCN and GAT is aggregation coefficients. Meanwhile, GCN and GAT use a 1-layer MLP in the structural/feature extraction step while GIN uses a 2-layer MLP.

In this three-stage representation, the aggregation is reformulated as the multiplication of the aggregation coefficients vector and the feature matrix of neighbors. Next, we explain how to characterize the permutation invariance in this formulation. Let $\boldsymbol{M} \in \mathbb{R}^{s \times n}$ denote an aggregation coefficient matrix where $s \geq 1$. Note that in GCN, GIN and GAT, s is restricted to be 1. $\boldsymbol{h}_\pi \in \mathbb{R}^{n \times d}$ is the matrix representation of n input elements according to π. The aggregation computation in the second step of Equation 2 is $\boldsymbol{f}_{\text{aggr}}(\boldsymbol{M}, \boldsymbol{h}_\pi) = \boldsymbol{M}\boldsymbol{P}_\pi \boldsymbol{h}_\pi = \boldsymbol{M}_\pi \boldsymbol{h}_\pi$, where $\boldsymbol{P}_\pi$ is the permutation matrix according to π. $\boldsymbol{P}_\pi \boldsymbol{h}_\pi$ ensures the same output for all $\boldsymbol{h}_\pi, \pi \in \Pi_{|h|}$. $\boldsymbol{M}_\pi = \boldsymbol{M}\boldsymbol{P}_\pi$ is the reordering of columns of $\boldsymbol{M}$ according to π. For any $\pi_1, \pi_2 \in \Pi_n$, $\boldsymbol{f}_{\text{aggr}}(\boldsymbol{M}, \boldsymbol{h}_{\pi_1}) = \boldsymbol{f}_{\text{aggr}}(\boldsymbol{M}, \boldsymbol{h}_{\pi_2})$, thus permutation invariance holds. Once $\boldsymbol{M}$ is decided, we obtain a unique aggregator denoted by $\boldsymbol{f}_{\boldsymbol{M}}$. For any sequence of input elements $\boldsymbol{h}$, $\boldsymbol{f}_{\boldsymbol{M}}(\boldsymbol{h}) = \boldsymbol{f}_{\text{aggr}}(\boldsymbol{M}, \boldsymbol{h}_\pi)$, where $\pi \in \Pi_n$ can be any ordering of neighbors. Next, we analyze the distinguishing strength of $\boldsymbol{f}_{\boldsymbol{M}}$.

Proposition 1. *For any two matrices $\boldsymbol{M} \in \mathbb{R}^{s \times n}$ and $\boldsymbol{M}' \in \mathbb{R}^{s' \times n}$ with $s, s' \leq n$, we have (i) $\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)} \succeq \boldsymbol{f}_{\boldsymbol{M}}$, where $\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)$ means stacking these two matrices; (ii) $\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)} \succ \boldsymbol{f}_{\boldsymbol{M}}$ if and only if rank$(\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)) >$ rank$(\boldsymbol{M})$; (iii) Any multiset of size n is distinguishable with $\boldsymbol{f}_{\boldsymbol{M}}$ if and only if rank$(\boldsymbol{M}) = n$.*

We prove Proposition 1 in Appendix C. Proposition 1 shows that the distinguishing strength of an aggregator is decided by the rank of the corresponding $\boldsymbol{M}$. Yet, Proposition 1 is only suiable for multisets aggregated with shared $\boldsymbol{f}_{\boldsymbol{M}}$. The aggregation conducted on different nodes may use different $\boldsymbol{M}$, also known as different aggregators, such as in GCN, GIN and GAT.

To maximally distinguish different structures, the aggregation results using different aggregators should be different no matter what the inputs of aggregators are. This is because the aggregation results are the mapping of local structures. For example, the aggregation results of a k-layer GNN are the mapping of k-hop neighbors assigned on each node. To maximally distinguish different local structures, the aggregation operation should map them to different results. Note that the aggregation coefficients matrix $\boldsymbol{M}$ is limited to be the mapping of local structures as given in Equation 2, which means different $\boldsymbol{M}$ corresponds to different local structures. Therefore, the aggregation results of different aggregators must be different. However, it is not satisfied by existing GNNs, and there are few studies on distinguishing multisets aggregated by different aggregators. In Proposition 2, we present a detailed analysis of it. Let Res$(\boldsymbol{f}_{\boldsymbol{M}})$ denote the set of results aggregated by $\boldsymbol{f}_{\boldsymbol{M}}$.

Proposition 2. *For any $\boldsymbol{M}_1, \boldsymbol{M}_2 \in \mathbb{R}^{s \times n_1}$ and $\boldsymbol{M}_1', \boldsymbol{M}_2' \in \mathbb{R}^{s' \times n_2}$, (i) Res$(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}_1' \end{smallmatrix}\right)}) \cap$ Res$(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_2 \\ \boldsymbol{M}_2' \end{smallmatrix}\right)}) \subseteq$ Res$(\boldsymbol{f}_{\boldsymbol{M}_1}) \cap$ Res$(\boldsymbol{f}_{\boldsymbol{M}_2})$; (ii) If Res$(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}_1' \end{smallmatrix}\right)}) \cap$ Res$(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_2 \\ \boldsymbol{M}_2' \end{smallmatrix}\right)}) \subset$ Res$(\boldsymbol{f}_{\boldsymbol{M}_1}) \cap$ Res$(\boldsymbol{f}_{\boldsymbol{M}_2})$, then rank$(\left(\begin{smallmatrix} \boldsymbol{M}_1 & \boldsymbol{M}_2 \\ \boldsymbol{M}_1' & \boldsymbol{M}_2' \end{smallmatrix}\right)) >$ rank$(\left(\begin{smallmatrix} \boldsymbol{M}_1 & \boldsymbol{M}_2 \end{smallmatrix}\right))$;*

We prove Proposition 2 in Appendix D. Next, we provide a sufficient condition for building desired multiple injective aggregators with the outputs having no intersections.

Proposition 3. *For any two aggregators $\boldsymbol{f}_{\boldsymbol{M}_1}$ and $\boldsymbol{f}_{\boldsymbol{M}_2}$ with $\boldsymbol{M}_1 \in \mathbb{R}^{s\times n_1}$ and $\boldsymbol{M}_2 \in \mathbb{R}^{s\times n_2}$, if rank$(\begin{pmatrix} \boldsymbol{M}_1 & \boldsymbol{M}_2 \end{pmatrix}) = n_1 + n_2$, then $\boldsymbol{f}_{\boldsymbol{M}_1}$ and $\boldsymbol{f}_{\boldsymbol{M}_2}$ are injective and Res$(\boldsymbol{f}_{\boldsymbol{M}_1}) \cap$ Res$(\boldsymbol{f}_{\boldsymbol{M}_2}) = \emptyset$.*

We prove Proposition 3 in Appendix E. Proposition 1, 2 and 3 provide a new perspective for building powerful aggregators and even injective aggregators. Compared with the distinguishing strength studies in Xu et al. (2019) and Corso et al. (2020), as well as existing strategies for building injective aggregators, e.g., sum-of-power mapping Zaheer et al. (2017) and normalized moments Corso et al. (2020), we reformulate the aggregation with aggregation coefficients matrix and show the relations of the distinguishing strength of aggregators and the rank of the corresponding aggregation coefficients matrices. Besides, the aggregation of this method is controlled by aggregation coefficients which can be learned from graph data to better leverage structural information. In this paper, to simplify the analysis, we only consider the aggregations within one-hop neighbors. The results can be easily extended to more sophisticated aggregators with the overall framework unchanged

In the perspective of preserving the rank of hidden features among neighbors, $\boldsymbol{r} = \boldsymbol{M}\boldsymbol{h}$ indicates that rank$(\boldsymbol{r}) < \min(\text{rank}(\boldsymbol{M}), \text{rank}(\boldsymbol{h}))$. To preserve the rank of hidden features in aggregations such that rank$(\boldsymbol{r}) =$ rank$(\boldsymbol{h})$, we need rank$(\boldsymbol{M}) \geq$ rank$(\boldsymbol{h})$. This builds a connection between improving the distinguishing strength of aggregators and preserving the rank of hidden features among neighbors, both of which have the requirements on the rank of $\boldsymbol{M}$. General aggregators such as ones in GCN, GIN-0 and GAT have rank$(\boldsymbol{M}) = 1$. Thus, rank$(\boldsymbol{r})$ is always fixed to 1 no matter what the rank of the input features is. Correspondingly, they have a weak distinguishing strength.

Equation 2 splits the aggregation and feature/structure extraction into two independent steps, which helps to figure out that the expressive power loss happens in the aggregation step, and then the model extracts feature/structure information on the distorted encodings of neighbors. From Equation 2, the aggregation can be considered as a representation regularization step, which unifies different multisets of neighbors into the same representation style while holding permutation invariance. Then, the model can extract structural information on this regulated data with a shared trainable matrix as the third step in Equation 2. Based on this observation, we propose two novel GNN layers: ExpandingConv and CombConv.

3.3 EXPANDINGCONV

We first give ExpandingConv framework. Then we provide one of its implementations and analyze how ExpandingConv achieves more powerful aggregations. The ExpandingConv framework is

$$\begin{cases} \boldsymbol{m}_{uv}^{(t)} = \boldsymbol{f}_{\text{local}}(u,v)|_{u\in\mathcal{N}(v)} \\ \boldsymbol{h}_v^{(t)} = \boldsymbol{f}_{\text{aggr}}(\{\{\text{vec}(\boldsymbol{m}_{uv}^{(t)}\boldsymbol{h}_u^{(t-1)T})|u\in\mathcal{N}(v)\}\}), \end{cases}$$

where $\boldsymbol{m}_{uv}^{(t)} \in \mathbb{R}^{s\times 1}$ with $s > 1$ and $\boldsymbol{f}_{\text{local}}(u,v)$ is the mapping of local structures between nodes u and v. The implementation of $\boldsymbol{f}_{\text{local}}(u,v)$ is very flexible with the only restriction of ensuring the same encoding of isomorphic graphs. vec$(\boldsymbol{m}_{uv}^{(t)}\boldsymbol{h}_u^{(t-1)T}) \in \mathbb{R}^{sd\times 1}$ is the expanded representation of hidden features $h_u^{(t-1)} \in \mathbb{R}^{d\times 1}$. Then a GNN layer $\boldsymbol{f}_{\text{aggr}}$ learns structural information on this expanded representations. We introduce an implelentation as follows:

$$\begin{cases} \boldsymbol{m}_{uv}^{(t)} = \text{Tanh}(\boldsymbol{W}[\boldsymbol{h}_v^{(t-1)}||\boldsymbol{h}_u^{(t-1)}] + \boldsymbol{b}) \\ \boldsymbol{h}_v^{(t)} = \sum_{u\in\mathcal{N}(v)} \text{MLP}(\text{vec}(\boldsymbol{m}_{uv}^{(t)}\boldsymbol{h}_u^{(t-1)T})) \end{cases} \tag{3}$$

In Equation 3, we implement $\boldsymbol{f}_{\text{local}}(u,v)$ as the function of hidden features of nodes u and v. There can be other implementations, and we leave them for future work. $\boldsymbol{W} \in \mathbb{R}^{s\times 2d}$ and $\boldsymbol{b} \in \mathbb{R}^{s\times 1}$ are trainable matrices. We use Tanh as the activation function in the computation of $\boldsymbol{m}_{uv}^{(t)}$ to preserve the rank of aggregation coefficient matrices Luan et al. (2019). MLP denotes a 2-layer perceptron.

Next, we represent Equation 3 with the corresponding three-stage representation as given in Section 3.2 to obtain its aggregation coefficient matrix and analyze its distinguishing strength. To simplify

this process, we only consider 1-layer MLP with $\boldsymbol{W}' \in \mathbb{R}^{d\times sd}$ and $\boldsymbol{b}' \in \mathbb{R}^{d\times 1}$.

$$
\begin{aligned}
&\boldsymbol{h}_v^{(t)} = \sum_{u\in\mathcal{N}(v)} \mathrm{ReLU}(\boldsymbol{W}'\mathrm{vec}(\boldsymbol{m}_{uv}^{(t)}\boldsymbol{h}_u^{(t-1)T}) + \boldsymbol{b}') = \\
&\begin{pmatrix} \sum_{u\in\mathcal{N}(v)} \mathrm{ReLU}(\boldsymbol{W}'_{[1,:]}\mathrm{vec}(\boldsymbol{m}_{uv}^{(t)}\boldsymbol{h}_u^{(t-1)T}) + \boldsymbol{b}'_{[1]}) \\ \sum_{u\in\mathcal{N}(v)} \mathrm{ReLU}(\boldsymbol{W}'_{[2,:]}\mathrm{vec}(\boldsymbol{m}_{uv}^{(t)}\boldsymbol{h}_u^{(t-1)T}) + \boldsymbol{b}'_{[2]}) \\ \vdots \\ \sum_{u\in\mathcal{N}(v)} \mathrm{ReLU}(\boldsymbol{W}'_{[d,:]}\mathrm{vec}(\boldsymbol{m}_{uv}^{(t)}\boldsymbol{h}_u^{(t-1)T}) + \boldsymbol{b}'_{[d]}) \end{pmatrix} = \begin{pmatrix} \sum_{u\in\mathcal{N}_1(v)} (\boldsymbol{W}'_{[1,:]}\mathrm{vec}(\boldsymbol{m}_{uv}^{(t)}\boldsymbol{h}_u^{(t-1)T}) + \boldsymbol{b}'_{[1]}) \\ \sum_{u\in\mathcal{N}_2(v)} (\boldsymbol{W}'_{[2,:]}\mathrm{vec}(\boldsymbol{m}_{uv}^{(t)}\boldsymbol{h}_u^{(t-1)T}) + \boldsymbol{b}'_{[2]}) \\ \vdots \\ \sum_{u\in\mathcal{N}_d(v)} (\boldsymbol{W}'_{[d,:]}\mathrm{vec}(\boldsymbol{m}_{uv}^{(t)}\boldsymbol{h}_u^{(t-1)T}) + \boldsymbol{b}'_{[d]}) \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{W}'_{[1,:]} & & & \\ & \boldsymbol{W}'_{[2,:]} & & \\ & & \ddots & \\ & & & \boldsymbol{W}'_{[d,:]} \end{pmatrix} \begin{pmatrix} \mathrm{vec}(\sum_{u\in\mathcal{N}_1(v)} \boldsymbol{m}_{uv}^{(t)}\boldsymbol{h}_u^{(t-1)T}) \\ \mathrm{vec}(\sum_{u\in\mathcal{N}_2(v)} \boldsymbol{m}_{uv}^{(t)}\boldsymbol{h}_u^{(t-1)T}) \\ \vdots \\ \mathrm{vec}(\sum_{u\in\mathcal{N}_d(v)} \boldsymbol{m}_{uv}^{(t)}\boldsymbol{h}_u^{(t-1)T}) \end{pmatrix} + \begin{pmatrix} |\mathcal{N}_1(v)|\cdot\boldsymbol{b}'_{[1]} \\ |\mathcal{N}_2(v)|\cdot\boldsymbol{b}'_{[2]} \\ \vdots \\ |\mathcal{N}_d(v)|\cdot\boldsymbol{b}'_{[d]} \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{W}'_{[1,:]} & & & \\ & \boldsymbol{W}'_{[2,:]} & & \\ & & \ddots & \\ & & & \boldsymbol{W}'_{[d,:]} \end{pmatrix} \begin{pmatrix} \mathrm{vec}(\boldsymbol{M}_{v_1\pi_1}^{(t)}\bar{\boldsymbol{h}}_{v_1\pi_1}^{(t-1)T}) \\ \mathrm{vec}(\boldsymbol{M}_{v_2\pi_2}^{(t)}\bar{\boldsymbol{h}}_{v_2\pi_2}^{(t-1)T}) \\ \vdots \\ \mathrm{vec}(\boldsymbol{M}_{v_d\pi_d}^{(t)}\bar{\boldsymbol{h}}_{v_d\pi_d}^{(t-1)T}) \end{pmatrix} + \begin{pmatrix} |\mathcal{N}_1(v)|\cdot\boldsymbol{b}'_{[1]} \\ |\mathcal{N}_2(v)|\cdot\boldsymbol{b}'_{[2]} \\ \vdots \\ |\mathcal{N}_d(v)|\cdot\boldsymbol{b}'_{[d]} \end{pmatrix},
\end{aligned}
\tag{4}
$$

where $\mathcal{N}_i(v) \subseteq \mathcal{N}(v)|i \in [d]$ are sampled subsets of neighbors in each dimension. $\boldsymbol{M}_{v_i}^{(t)} = (\boldsymbol{m}_{u_1v}^{(t)}, \boldsymbol{m}_{u_2v}^{(t)}, \cdots, \boldsymbol{m}_{u_{|\mathcal{N}_i(v)|}v}^{(t)}) \in \mathbb{R}^{s\times|\mathcal{N}_i(v)|}$ and $\bar{\boldsymbol{h}}_{v_i}^{(t-1)} = (\boldsymbol{h}_{u_1}^{(t-1)}, \boldsymbol{h}_{u_2}^{(t-1)}, \cdots, \boldsymbol{h}_{u_{|\mathcal{N}_i(v)|}}^{(t-1)}) \in \mathbb{R}^{d\times|\mathcal{N}_i(v)|}$ are aggregation coefficients matrix and hidden feature matrix corresponding to the subset of neighbors $\mathcal{N}_i(v) \subseteq \mathcal{N}(v)$ according to π_i. We denote $[\boldsymbol{h}_v^{(t-1)}||\bar{\boldsymbol{h}}_{v_i}^{(t-1)}] = (\boldsymbol{h}_v^{(t-1)}||\boldsymbol{h}_{u_1}^{(t-1)}, \boldsymbol{h}_v^{(t-1)}||\boldsymbol{h}_{u_2}^{(t-1)}, \cdots, \boldsymbol{h}_v^{(t-1)}||\boldsymbol{h}_{u_{|\mathcal{N}_i(v)|}}^{(t-1)}) \in \mathbb{R}^{2d\times|\mathcal{N}_i(v)|}$, then $\boldsymbol{M}_{v_i}^{(t)} = \mathrm{Tanh}(\boldsymbol{W}[\boldsymbol{h}_v^{(t-1)}||\bar{\boldsymbol{h}}_{v_i}^{(t-1)}] + \boldsymbol{b}) \in \mathbb{R}^{s\times|\mathcal{N}_i(v)|}$. According to Equation 4, we finally obtain the three-stage representation equivalent to Equation 3.

$$
\begin{cases}
\boldsymbol{M}_{v_i}^{(t)} = \mathrm{Tanh}(\boldsymbol{W}[\boldsymbol{h}_v^{(t-1)}||\bar{\boldsymbol{h}}_{v_i}^{(t-1)}] + \boldsymbol{b}) \\
\bar{\boldsymbol{r}}_{v_i}^{(t)} = \boldsymbol{M}_{v_i\pi_i}^{(t)}\bar{\boldsymbol{h}}_{v_i\pi_i}^{(t-1)T}, \quad i \in [d] \\
\boldsymbol{h}_v^{(t)} = \mathrm{diag}(\boldsymbol{W}'_{[1,:]}, \boldsymbol{W}'_{[2,:]}, \cdots, \boldsymbol{W}'_{[d,:]})(\mathrm{vec}(\bar{\boldsymbol{r}}_{v_1}^{(t)}), \mathrm{vec}(\bar{\boldsymbol{r}}_{v_2}^{(t)}), \cdots, \mathrm{vec}(\bar{\boldsymbol{r}}_{v_d}^{(t)}))^T \\
\qquad +(|\mathcal{N}_1(v)|\cdot\boldsymbol{b}'_{[1]}, |\mathcal{N}_2(v)|\cdot\boldsymbol{b}'_{[2]}, \cdots, |\mathcal{N}_d(v)|\cdot\boldsymbol{b}'_{[d]})^T.
\end{cases}
\tag{5}
$$

According to the computation of $\bar{\boldsymbol{r}}_{v_i}^{(t)}$, $\mathrm{rank}(\bar{\boldsymbol{r}}_{v_i}^{(t)}) \leq \min(s, d, |\mathcal{N}_i(v)|)$. By configuring a larger s, we have $\mathrm{rank}(\bar{\boldsymbol{r}}_{v_i}^{(t)}) > 1$ with a high probability, which is different from general GNNs with a rank of 1. As analyzed in Section 3.2, this achieves more powerful aggregators as well as preserves the rank of hidden features among neighbors. The obtained $\bar{\boldsymbol{r}}_{v_i}^{(t)}$ after aggregation is the unified representations of neighbors. We then use the trainable matrix $\boldsymbol{W}' \in \mathbb{R}^{sd\times d}$ to extracts feature/structure information. Unlike the aggregation step, the dimensions reduction here (from sd to d) would not cause information loss. This can be explained that only task-relevant structural information needs to be preserved and passed to the next layer, and it can be embedded in lower dimensions.

Comparisons with multi-head GAT.

Proposition 4. *Multi-head GAT is an implementation of ExpandingConv as follows:*

$$
\begin{cases}
\boldsymbol{\alpha}_{vu} = \mathit{softmax}(\mathit{LeakyReLU}([\mathit{diag}(\tilde{\boldsymbol{a}}^{1T}, \tilde{\boldsymbol{a}}^{2T}, \cdots, \tilde{\boldsymbol{a}}^{KT}) \\
\qquad ||\mathit{diag}(\tilde{\boldsymbol{a}}'^{1T}, \tilde{\boldsymbol{a}}'^{2T}, \cdots, \tilde{\boldsymbol{a}}'^{KT})][\boldsymbol{W}\boldsymbol{h}_v^{(t-1)}||\boldsymbol{W}\boldsymbol{h}_u^{(t-1)}])) \\
\boldsymbol{h}_v^{(t)} = \sigma\left(\frac{1}{K}\boldsymbol{W}\sum_{j\in\mathcal{N}_v} \mathit{vec}(\boldsymbol{\alpha}_{vu}\boldsymbol{h}_u^{(t-1)T})\right),
\end{cases}
\tag{6}
$$

where $\boldsymbol{W} = ||_{k=1}^{K}\boldsymbol{W}^k \in \mathbb{R}^{kd\times d}$ is the concatenation of the trainable matrix in all K heads.

We prove Proposition 4 in Appendix F. Although multi-head GAT is based on attention mechanism, ExpandingConv provides a new perspective to explain its effectiveness. Applying multi-head attention mechanism helps to preserve the rank of hidden features as well as achieve more powerful aggregators, leading to less distortion among neighbors.

GAT as well as most other GNNs (such as GCN, GIN, etc) follows the same pattern that applies nonlinear units after aggregation. According to the analysis in Section 3.1, Equation 3 applies MLP on $\text{vec}(\boldsymbol{m}_{uv}^{(t)}\boldsymbol{h}_u^{(t-1)T})$ before SUM to break the distinguishing strength limitation of SUM. It also produces other interesting results. By reformulating Equation 3 with its three-stage representation as Equation 5, each dimension of hidden features aggregates on a subset of neighbors independently, which corresponds to a kind of dimension-wise neighbor sampling mechanism. We call the modification of applying ReLU ahead of SUM aggregator as Re-SUM mechanism. Mishra et al. (2020) and Rong et al. (2019) studied dropedge and node masking mechanism on node-level predictions, both of which can be considered as neighbor sampling strategies that have shown their effectiveness in improving the generalization ability of aggregation-based GNNs and are also used as unbiased data augmentation technique for training. Compared with dropedge and node masking, Re-SUM realizes a dimension-wise neighbor sampling. and it does not need to manually set the sampling ratio since this mechanism takes effects implicitly. Re-SUM shows that the neural network itself can perform sampling by properly combining nonlinear units and aggregators, without explicitly modifying the network architecture. Our experimental results verified the effectiveness of the Re-SUM on a variety of graph tasks.

3.4 COMBCONV

The CombConv framework is

$$\begin{cases} \boldsymbol{m}_{uv}^{(t)} = \boldsymbol{f}_{\text{local}}(u,v)|_{u\in\mathcal{N}(v)} \\ \boldsymbol{h}_v^{(t)} = \boldsymbol{f}_{\text{aggr}}(\{\{\text{vec}(\boldsymbol{m}_{uv}^{(t)} \odot \boldsymbol{h}_u^{(t-1)})|u\in\mathcal{N}(v)\}\}), \end{cases}$$

where $\boldsymbol{m}_{uv}^{(t)} \in \mathbb{R}^{d\times 1}$ and $\odot$ denotes element-wise product. An implementation of CombConv is given as follows:

$$\begin{cases} \boldsymbol{m}_{uv}^{(t)} = \text{Tanh}(\boldsymbol{W}[\boldsymbol{h}_v^{(t-1)}||\boldsymbol{h}_u^{(t-1)}] + \boldsymbol{b}) \\ \boldsymbol{h}_v^{(t)} = \sum_{u\in\mathcal{N}(v)} \text{MLP}(\boldsymbol{m}_{uv}^{(t)} \odot \boldsymbol{h}_u^{(t-1)}), \end{cases} \tag{7}$$

where $\boldsymbol{W} \in \mathbb{R}^{d\times 2d}$ and $\boldsymbol{b} \in \mathbb{R}^{d\times 1}$. Similar to ExpandingConv, CombConv also applies Re-SUM aggregation. The difference is that each dimension of hidden features is aggregated with an independent weighted aggregator, which works like a comb. ExpandingConv with $s = 1$ corresponds to a special case of CombConv where all dimensions share the same aggregator. Therefore, the distinguishing strength of CombConv is stronger than ExpandingConv with $s = 1$. Meanwhile, CombConv does not expand the hidden features of nodes in aggregation. Hence, it requires fewer parameters.

4 EXPERIMENTS

In this section, we evaluate ExpandingConv and CombConv on graph-level prediction tasks on OGB Weihua Hu (2020) and QM9 Ramakrishnan et al. (2014); Wu et al. (2018); Ruddigkeit et al. (2012). All datasets' descriptions and detailed experimental setup are given in Appendix H. The code is available at `https://github.com/qslim/epcb-gnns`.

We first conduct comprehensive ablation studies to evaluate the effectiveness of ExpandingConv, CombConv and Re-SUM mechanism as given in Table 1. ExpC-s denotes ExpandingConv with $\boldsymbol{W} \in \mathbb{R}^{s\times\star}$. We use ExpC* and CombC* to denote the ExpandingConv and CombConv without Re-SUM.

Effect of powerful aggregators. For complex graphs structures with dense connections or with abundant node/edge features, they would benefit from a higher expressive model to maximumly distinguish different structures and extract relevant structural patterns as the model goes deeper to leverage large receptive fields. This is validated on both QM9 and OGB. We config $s = 1, 4, 8, 16, 32$ of ExpC-s for all 12 targets of QM9. As we apply larger s, the model continuously achieves better performance on most targets. We randomly select $s = 1, 4$ for ogbg-ppa, ogbg-molhiv and $s = 1, 5$ for ogbg-code. The results show that applying larger s gains performance improvements, especially on ogbg-ppa which involves large graphs with dense connections.

Effect of Re-SUM mechanism. In Table 1, the performance differences between ExpC*-1 (CombC*) and ExpC-1 (CombC) show the effectiveness of Re-SUM. In our tests, the Re-SUM

Table 1: Ablation studies on OGB and QM9. Higher is better for OGB and lower is better for QM9.

	ogbg-ppa		ogbg-molhiv		ogbg-code	
	Test Accuracy	Validation Accuracy	Test ROC-AUC	Validation ROC-AUC	Test F1 score	Validation F1 score
ExpC*-1	0.7065	0.6753	0.7763	0.8082	0.322	0.303
ExpC-1	0.7750	0.7334	0.7679	0.8025	0.326	0.3076
ExpC-4,5	**0.8011**	**0.7504**	**0.7789**	**0.8120**	**0.332**	**0.3114**
CombC*	0.7361	0.7109	0.7647	0.8119	0.3229	0.3026
CombC	0.7764	0.7243	0.7663	0.7946	0.3272	0.3064

	μ	α	ϵ_{homo}	ϵ_{lumo}	$\Delta\epsilon$	$\langle R^2 \rangle$	$ZPVE$	U_0	U	H	G	C_v
ExpC*-1	0.467	0.283	0.00337	0.00340	0.00467	22.9	0.000205	0.0255	0.0263	0.0242	0.0261	0.1189
ExpC-1	0.469	0.268	0.00326	0.00329	0.00466	20.8	0.000186	0.0202	0.0199	0.0202	0.0201	0.1039
ExpC-4	0.413	0.255	0.00273	0.00300	0.00420	19.4	**0.000168**	0.0184	0.0183	0.0178	0.0182	0.1115
ExpC-8	0.400	0.257	0.00259	0.00286	0.00395	18.1	0.000172	0.0158	0.0170	0.0177	0.0184	0.1060
ExpC-16	0.382	0.255	**0.00248**	0.00268	0.00373	17.2	0.000170	0.0170	0.0174	0.0193	**0.0165**	0.1043
ExpC-32	0.368	**0.244**	**0.00248**	**0.00257**	**0.00364**	16.3	0.000174	**0.0151**	**0.0167**	**0.0165**	0.0198	0.0962
CombC*	0.4062	0.248	0.00259	0.00273	0.00387	17.1	0.000170	0.0185	0.0181	0.0164	0.0174	0.1022
CombC	0.399	**0.241**	0.00261	0.00278	0.00386	**15.9**	**0.000160**	**0.0144**	**0.0145**	**0.0147**	**0.0140**	**0.0858**

can be extremely powerful on graphs with dense connections such as ogbg-ppa, which is validated on both ExpandingConv (with 6.85% improvements) and CombConv (with 4% improvements). On most targets of QM9, this mechanism also gains improvements.

Table 2: Comparisons with baselines on OGB and QM9.

	ogbg-ppa		ogbg-molhiv		ogbg-code	
	Test Accuracy	Validation Accuracy	Test ROC-AUC	Validation ROC-AUC	Test F1 score	Validation F1 score
GCN Kipf & Welling (2016)	0.6839 ± 0.0084	0.6497 ± 0.0034	0.7606 ± 0.0097	0.8204 ± 0.0141	0.3163 ± 0.0018	0.2973 ± 0.0014
GIN Xu et al. (2019)	0.6892 ± 0.0100	0.6562 ± 0.0107	0.7558 ± 0.0140	0.8232 ± 0.0090	0.3163 ± 0.0020	0.2981 ± 0.0014
DeeperGCN Li et al. (2020a)	0.7712 ± 0.0071	0.7313 ± 0.0078	0.7858 ± 0.0117	0.8427 ± 0.0063	NA	NA
HIMP Fey et al. (2020)	NA	NA	$\mathbf{0.7880 \pm 0.0082}$	NA	NA	NA
WEGL Kolouri et al. (2020)	NA	NA	0.7757 ± 0.0111	0.8101 ± 0.0097	NA	NA
ExpC-s	$\mathbf{0.7976 \pm 0.0072}$	$\mathbf{0.7518 \pm 0.0080}$	0.7799 ± 0.0082	0.7940 ± 0.0135	$\mathbf{0.3318 \pm 0.0017}$	$\mathbf{0.3119 \pm 0.0008}$
CombC	0.7781 ± 0.0076	0.7253 ± 0.0092	0.7715 ± 0.0132	0.7968 ± 0.0125	0.3276 ± 0.0015	0.3061 ± 0.0011

	μ	α	ϵ_{homo}	ϵ_{lumo}	$\Delta\epsilon$	$\langle R^2 \rangle$	$ZPVE$	U_0	U	H	G	C_v
DTNN Wu et al. (2018)	**0.244**	0.95	0.00388	0.00512	0.0112	17	0.00172	2.43	2.43	2.43	2.43	0.27
MPNN Gilmer et al. (2017)	0.358	0.89	0.00541	0.00623	0.0066	28.5	0.00216	2.05	2	2.02	2.02	0.42
k-GNN Morris et al. (2019)	0.476	0.27	0.00337	0.00351	0.0048	22.9	0.00019	0.0427	0.111	0.0419	0.0469	**0.0944**
PPGN Maron et al. (2019)	**0.0934**	0.318	**0.00174**	**0.0021**	**0.0029**	**3.78**	0.000399	0.022	0.0504	0.0294	0.024	0.144
GIN0* Xu et al. (2019)	0.471	0.281	0.00327	0.00340	0.00473	22.9	0.000202	0.0244	0.0245	0.0233	0.0255	0.1283
ExpC-s	0.368	**0.244**	**0.00248**	**0.00257**	**0.00364**	16.3	**0.000168**	**0.0151**	**0.0167**	**0.0165**	**0.0165**	0.0962
CombC	0.399	**0.241**	0.00261	0.00278	0.00386	**15.9**	**0.000160**	**0.0144**	**0.0145**	**0.0147**	**0.0140**	**0.0858**

Comparisons with baselines. Table 2 shows the performance comparisons of our models with baselines evaluated on QM9 and OGB respectively. All results of baselines are taken from the original papers except for GIN0 on QM9. To ensure a fair comparison, we implement GIN0 without batch normalization denoted by GIN0*. For ExpandingConv, we set $s = 4$ for ogbg-ppa and ogbg-molhiv; $s = 5$ for ogbg-code; $s \in \{4, 8, 16, 32\}$ for all 12 targets in QM9. Note that our models achieve the lowest error on 7 out of the 12 targets of QM9. On ogbg-ppa, ExpandingConv and CombConv achieve 2.6% and 0.7% higher classification accuracies respectively compared with SOTA models. On ogbg-code, they achieve 1.5% and 1.1% improvements.

5 Conclusion

We show how basic aggregators used in general GNNs become expressive bottlenecks. To address this limitation, we develop theoretical foundations of building powerful aggregators. We also propose the Re-SUM mechanism which achieves dimension-wise sampling. To evaluate their effectiveness, we develop two novel GNN layers, and conduct extensive experiments on public graph benchmarks. The results are consistent with our analysis, and our proposed models achieve SOTA performance on a variety of graph-level prediction benchmarks.

REFERENCES

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. In *Advances in Neural Information Processing Systems*, pp. 15868–15876, 2019.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *arXiv preprint arXiv:2004.05718*, 2020.

Nima Dehmamy, Albert-László Barabási, and Rose Yu. Understanding the representation power of graph neural networks in learning graph topology. In *Advances in Neural Information Processing Systems*, pp. 15413–15423, 2019.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

M. Fey, J. G. Yuen, and F. Weichert. Hierarchical inter-message passing for learning on molecular graphs. In *ICML Graph Representation Learning and Beyond (GRL+) Workhop*, 2020.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1263–1272. JMLR. org, 2017.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734. IEEE, 2005.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. Benchmark data sets for graph kernels, 2016. `http://graphkernels.cs.tu-dortmund.de`.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Soheil Kolouri, Navid Naderializadeh, Gustavo K Rohde, and Heiko Hoffmann. Wasserstein embedding for graph learning. *arXiv preprint arXiv:2006.09430*, 2020.

Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020a.

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding–design provably more powerful gnns for structural representation learning. *arXiv preprint arXiv:2009.00142*, 2020b.

Yaw-Ling Lin and Steven S Skiena. Algorithms for square roots of graphs. *SIAM Journal on Discrete Mathematics*, 8(1):99–118, 1995.

Sitao Luan, Mingde Zhao, Xiao-Wen Chang, and Doina Precup. Break the ceiling: Stronger multi-scale deep graph convolutional networks. In *Advances in neural information processing systems*, pp. 10945–10955, 2019.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *arXiv preprint arXiv:1905.11136*, 2019.

Pushkar Mishra, Aleksandra Piktus, Gerard Goossen, and Fabrizio Silvestri. Node masking: Making graph neural networks generalize and scale better. *arXiv preprint arXiv:2001.07524*, 2020.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

Ryan L Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. *arXiv preprint arXiv:1811.01900*, 2018.

Ryan L Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational pooling for graph representations. *arXiv preprint arXiv:1903.02541*, 2019.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1:140022, 2014.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2019.

Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.

Ryoma Sato. A survey on the expressive power of graph neural networks. *arXiv preprint arXiv:2003.04078*, 2020.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Clement Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with message-passing. *arXiv preprint arXiv:2006.15107*, 2020.

Marinka Zitnik Yuxiao Dong Hongyu Ren Bowen Liu Michele Catasta Jure Leskovec Weihua Hu, Matthias Fey. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

B. Yu. Weisfeiler and A. A. Leman. Reduction of a graph to a canonical form and an algebra arising during this reduction. 1968.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. *arXiv preprint arXiv:1806.03536*, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryGs6iA5Km`.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pp. 3391–3401, 2017.

A GCN, GAT AND GIN

Here, we present implementations of GCN, GAT and GIN for the usage of our analysis.

Graph Convolution Networks (GCN) Kipf & Welling (2016).

$$\boldsymbol{H}^{(t)} = \sigma(\hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(t-1)}\boldsymbol{W} + \boldsymbol{b}) \tag{8}$$

Graph Attention Networks (GAT) Veličković et al. (2017).

$$\boldsymbol{h}_v^{(t)} = \sigma\left(\sum_{u\in\mathcal{N}(v)} \text{att}\left(\boldsymbol{h}_v^{(t-1)}, \boldsymbol{h}_u^{(t-1)}\right)\boldsymbol{W}\boldsymbol{h}_u^{(t-1)}\right) \tag{9}$$

Graph Isomorphism Networks (GIN-0) Xu et al. (2019).

$$\boldsymbol{h}_v^{(t)} = \text{MLP}(\sum_{u\in\mathcal{N}(v)} \boldsymbol{h}_u^{(t-1)}). \tag{10}$$

B PROOF OF LEMMA 1

Proof. (i) For any two multisets $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, if $g(\boldsymbol{f}_{\text{aggr}}(\boldsymbol{x}_1)) \neq g(\boldsymbol{f}_{\text{aggr}}(\boldsymbol{x}_2))$, then $\boldsymbol{f}_{\text{aggr}}(\boldsymbol{x}_1) \neq \boldsymbol{f}_{\text{aggr}}(\boldsymbol{x}_2)$. Therefore, we have $\boldsymbol{f}_{\text{aggr}} \succeq g \circ \boldsymbol{f}_{\text{aggr}}$. If g is injective, then $\boldsymbol{f}_{\text{aggr}}(\boldsymbol{x}_1) \neq \boldsymbol{f}_{\text{aggr}}(\boldsymbol{x}_2) \Rightarrow g(\boldsymbol{f}_{\text{aggr}}(\boldsymbol{x}_1)) \neq g(\boldsymbol{f}_{\text{aggr}}(\boldsymbol{x}_2))$. We have $\boldsymbol{f}_{\text{aggr}} \succeq g \circ \boldsymbol{f}_{\text{aggr}} \succeq \boldsymbol{f}_{\text{aggr}}$, therefore $\boldsymbol{f}_{\text{aggr}} = g \circ \boldsymbol{f}_{\text{aggr}}$.

(ii) For any two multisets $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, $\boldsymbol{f}_{aggr1}(\boldsymbol{x}_1) \neq \boldsymbol{f}_{aggr1}(\boldsymbol{x}_2) \Rightarrow [\boldsymbol{f}_{aggr1}(\boldsymbol{x}_1)||\boldsymbol{f}_{aggr2}(\boldsymbol{x}_1)] \neq [\boldsymbol{f}_{aggr1}(\boldsymbol{x}_2)||\boldsymbol{f}_{aggr2}(\boldsymbol{x}_2)]$. Therefore, $\boldsymbol{f}_{aggr1} \otimes \boldsymbol{f}_{aggr2} \succeq \boldsymbol{f}_{aggr1}$. If $\boldsymbol{f}_{aggr1}$ and $\boldsymbol{f}_{aggr2}$ are incomparable, there exist $\boldsymbol{x}_3$ and $\boldsymbol{x}_4$ such that $\boldsymbol{f}_{aggr2}(\boldsymbol{x}_3) \neq \boldsymbol{f}_{aggr2}(\boldsymbol{x}_4)$ but $\boldsymbol{f}_{aggr1}(\boldsymbol{x}_3) = \boldsymbol{f}_{aggr1}(\boldsymbol{x}_4)$. Therefore, there exist $\boldsymbol{x}_3$ and $\boldsymbol{x}_4$ such that $[\boldsymbol{f}_{aggr1}(\boldsymbol{x}_3)||\boldsymbol{f}_{aggr2}(\boldsymbol{x}_3)] \neq [\boldsymbol{f}_{aggr1}(\boldsymbol{x}_4)||\boldsymbol{f}_{aggr2}(\boldsymbol{x}_4)]$ but $\boldsymbol{f}_{aggr1}(\boldsymbol{x}_3) = \boldsymbol{f}_{aggr1}(\boldsymbol{x}_4)$. $\boldsymbol{f}_{aggr1} \otimes \boldsymbol{f}_{aggr2} \succ \boldsymbol{f}_{aggr1}$.

(iii) Since $\boldsymbol{f}_{\text{aggr}}$ is an equivariant aggregator, then $\boldsymbol{f}_{\text{aggr}}(\boldsymbol{T}\cdot\boldsymbol{x}_1, \boldsymbol{T}\cdot\boldsymbol{x}_2, \cdots, \boldsymbol{T}\cdot\boldsymbol{x}_n) = \boldsymbol{T}\cdot\boldsymbol{f}_{\text{aggr}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n) \preceq \boldsymbol{f}_{\text{aggr}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)$. □

C PROOF OF PROPOSITION 1

Proof. (i)

$$\boldsymbol{f}_{\binom{M}{M'}}(\boldsymbol{x}) = \binom{M}{M'}_\pi \boldsymbol{x}_\pi = \begin{pmatrix}\boldsymbol{m}_\pi \boldsymbol{x}_\pi \\ \boldsymbol{m}'_\pi \boldsymbol{x}_\pi\end{pmatrix} = \begin{pmatrix}\boldsymbol{f}_{\boldsymbol{M}}(\boldsymbol{x}) \\ \boldsymbol{f}_{\boldsymbol{M}'}(\boldsymbol{x})\end{pmatrix},$$

then for any $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, we have

$$\boldsymbol{f}_{\boldsymbol{M}}(\boldsymbol{x}_1) \neq \boldsymbol{f}_{\boldsymbol{M}}(\boldsymbol{x}_2) \Rightarrow \boldsymbol{f}_{\binom{M}{M'}}(\boldsymbol{x}_1) \neq \boldsymbol{f}_{\binom{M}{M'}}(\boldsymbol{x}_2),$$

and therefore we conclude that $\boldsymbol{f}_{\binom{M}{M'}} \succeq \boldsymbol{f}_{\boldsymbol{M}}$.

(ii) "$\boldsymbol{f}_{\binom{M}{M'}} \succ \boldsymbol{f}_{\boldsymbol{M}} \Leftarrow \text{rank}(\binom{M}{M'}) > \text{rank}(\boldsymbol{M})$"

We prove the claim by contradiction. Assume that $\text{rank}(\binom{M}{M'}) > \text{rank}(\boldsymbol{M})$ and $\boldsymbol{f}_{\binom{M}{M'}} = \boldsymbol{f}_{\boldsymbol{M}}$. $\boldsymbol{f}_{\binom{M}{M'}} = \boldsymbol{f}_{\boldsymbol{M}}$ means that for any $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, $\boldsymbol{m}_\pi \boldsymbol{x}_{1\pi} = \boldsymbol{m}_\pi \boldsymbol{x}_{2\pi} \Leftrightarrow \binom{M}{M'}_\pi \boldsymbol{x}_{1\pi} = \binom{M}{M'}_\pi \boldsymbol{x}_{2\pi}$, where π is the ordering of input elements. Let $\boldsymbol{s} = \boldsymbol{x}_{1\pi} - \boldsymbol{x}_{2\pi}$. For any $i \in [n]$, $\boldsymbol{s}[i] = \boldsymbol{x}_{1\pi}[i] - \boldsymbol{x}_{2\pi}[i] \in \mathbb{R}$. Then for any $\boldsymbol{s} \in \mathbb{R}^n$, $\boldsymbol{m}_\pi \boldsymbol{s} = \boldsymbol{0} \Leftrightarrow \binom{M}{M'}_\pi \boldsymbol{s} = \boldsymbol{0}$. The system of linear equations $\boldsymbol{m}_\pi \boldsymbol{x} = \boldsymbol{0}$ and $\binom{M}{M'}_\pi \boldsymbol{x} = \boldsymbol{0}$ share the same solution space. Let R_S denote the rank of this solution space, then $\text{rank}(\boldsymbol{m}_\pi) + R_S = \text{rank}(\binom{M}{M'}_\pi) + R_S = n$. Therefore, $\text{rank}(\boldsymbol{m}_\pi) = \text{rank}(\binom{M}{M'}_\pi)$, then we have $\text{rank}(\boldsymbol{M}) = \text{rank}(\binom{M}{M'})$. Since we assumed that $\text{rank}(\binom{M}{M'}) > \text{rank}(\boldsymbol{M})$, we reach a contradiction.

"$\boldsymbol{f}_{\binom{M}{M'}} \succ \boldsymbol{f}_{\boldsymbol{M}} \Rightarrow \text{rank}(\binom{M}{M'}) > \text{rank}(\boldsymbol{M})$"

We prove an equivalent proposition "$\text{rank}(\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)) \leq \text{rank}(\boldsymbol{M}) \Rightarrow \boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)} \preceq \boldsymbol{f}_{\boldsymbol{M}}$". Note that $\text{rank}(\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)) \geq \text{rank}(\boldsymbol{M})$ and $\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)} \succeq \boldsymbol{f}_{\boldsymbol{M}}$ as given in Proposition 1(i). We only need to prove "$\text{rank}(\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)) = \text{rank}(\boldsymbol{M}) \Rightarrow \boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)} = \boldsymbol{f}_{\boldsymbol{M}}$". $\text{rank}(\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)) = \text{rank}(\boldsymbol{M})$ means that any row in $\boldsymbol{M}'$ is linearly dependent to rows in $\boldsymbol{M}$. Therefore, there exists $\boldsymbol{L} \in \mathbb{R}^{s' \times s}$ so that $\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right) = \left(\begin{smallmatrix} \boldsymbol{I} \\ \boldsymbol{L} \end{smallmatrix}\right)\boldsymbol{M}$. For any $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ with $\boldsymbol{M}\boldsymbol{P}_\pi\boldsymbol{x}_{1\pi} = \boldsymbol{M}\boldsymbol{P}_\pi\boldsymbol{x}_{2\pi}$, $\left(\begin{smallmatrix} \boldsymbol{I} \\ \boldsymbol{L} \end{smallmatrix}\right)\boldsymbol{M}\boldsymbol{P}_\pi\boldsymbol{x}_{1\pi} = \left(\begin{smallmatrix} \boldsymbol{I} \\ \boldsymbol{L} \end{smallmatrix}\right)\boldsymbol{M}\boldsymbol{P}_\pi\boldsymbol{x}_{2\pi}$, and therefore $\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)\boldsymbol{P}_\pi\boldsymbol{x}_{1\pi} = \left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)\boldsymbol{P}_\pi\boldsymbol{x}_{2\pi}$, where π is the ordering of input elements. That is, for any $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, $\boldsymbol{f}_{\boldsymbol{M}}(\boldsymbol{x}_1) = \boldsymbol{f}_{\boldsymbol{M}}(\boldsymbol{x}_2) \Rightarrow \boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)}(\boldsymbol{x}_1) = \boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)}(\boldsymbol{x}_2)$, thus $\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)} \preceq \boldsymbol{f}_{\boldsymbol{M}}$. Finally, we have $\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)} = \boldsymbol{f}_{\boldsymbol{M}}$.

(iii) "Any multiset of size n is distinguishable with $\boldsymbol{f}_{\boldsymbol{M}} \Rightarrow \text{rank}(\boldsymbol{M}) = n$"

Since $\text{rank}(\boldsymbol{M}) \leq n$, we prove an equivalent proposition "$\text{rank}(\boldsymbol{M}) < n \Rightarrow$ there exists at least two multisets which are indistinguishable". Considering the system of linear equations $\boldsymbol{y} = \boldsymbol{M}\boldsymbol{x}$ where $\boldsymbol{x} \in \mathbb{R}^n$, if $\text{rank}(\boldsymbol{M}) < n$, then there exists $\boldsymbol{y}'$ such that $\text{rank}(\boldsymbol{M}) = \text{rank}(\boldsymbol{M}, \boldsymbol{y}') < n$. According to the Rouché–Capelli theorem, there are infinite solutions $\boldsymbol{x}'_i$ such that $\boldsymbol{y}' = \boldsymbol{M}\boldsymbol{x}'_1 = \boldsymbol{M}\boldsymbol{x}'_2 = \cdots = \boldsymbol{M}\boldsymbol{x}'_\infty$, Each $\boldsymbol{x}'_i$ comes from a multiset with a particular order. Next, we need to prove that all these $\boldsymbol{x}'_i$ come from more than one multiset. As a multiset with bounded size n constitutes at most $n!$ different orders, the infinite number of $\boldsymbol{x}'_i$ corresponds to $\boldsymbol{y}'$ must come from more than one multisets, making these multisets indistinguishable.

"Any multiset of size n is distinguishable with $\boldsymbol{f}_{\boldsymbol{M}} \Leftarrow \text{rank}(\boldsymbol{M}) = n$"

Since $\text{rank}(\boldsymbol{M}) = n$ and $s = n$, for any $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} = \boldsymbol{M}\boldsymbol{x} \in \mathbb{R}^n$ is unique. Correspondingly, for any $\boldsymbol{P}_\pi\boldsymbol{x}_\pi$, $\boldsymbol{M}(\boldsymbol{P}_\pi\boldsymbol{x}_\pi)$ is unique.

□

D PROOF OF PROPOSITION 2

Proof. (i) According to the proof of Proposition 1(i), $\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M} \\ \boldsymbol{M}' \end{smallmatrix}\right)}(\boldsymbol{x}) = \begin{pmatrix} \boldsymbol{f}_{\boldsymbol{M}}(\boldsymbol{x}) \\ \boldsymbol{f}_{\boldsymbol{M}'}(\boldsymbol{x}) \end{pmatrix}$. For any $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ with $\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}'_1 \end{smallmatrix}\right)}(\boldsymbol{x}_1) = \boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_2 \\ \boldsymbol{M}'_2 \end{smallmatrix}\right)}(\boldsymbol{x}_2)$, $\boldsymbol{f}_{\boldsymbol{M}_1}(\boldsymbol{x}_1) = \boldsymbol{f}_{\boldsymbol{M}_2}(\boldsymbol{x}_2)$ holds. Meanwhile, $\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}'_1 \end{smallmatrix}\right)}(\boldsymbol{x}_1) = \boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_2 \\ \boldsymbol{M}'_2 \end{smallmatrix}\right)}(\boldsymbol{x}_2) \in \text{Res}(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}'_1 \end{smallmatrix}\right)}) \cap \text{Res}(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_2 \\ \boldsymbol{M}'_2 \end{smallmatrix}\right)})$, and $\boldsymbol{f}_{\boldsymbol{M}_1}(\boldsymbol{x}_1) = \boldsymbol{f}_{\boldsymbol{M}_2}(\boldsymbol{x}_2) \in \text{Res}(\boldsymbol{f}_{\boldsymbol{M}_1}) \cap \text{Res}(\boldsymbol{f}_{\boldsymbol{M}_2})$. Therefore, for any $e \in \text{Res}(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}'_1 \end{smallmatrix}\right)}) \cap \text{Res}(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_2 \\ \boldsymbol{M}'_2 \end{smallmatrix}\right)})$, we have $e \in \text{Res}(\boldsymbol{f}_{\boldsymbol{M}_1}) \cap \text{Res}(\boldsymbol{f}_{\boldsymbol{M}_2})$. That is $\text{Res}(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}'_1 \end{smallmatrix}\right)}) \cap \text{Res}(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_2 \\ \boldsymbol{M}'_2 \end{smallmatrix}\right)}) \subseteq \text{Res}(\boldsymbol{f}_{\boldsymbol{M}_1}) \cap \text{Res}(\boldsymbol{f}_{\boldsymbol{M}_2})$.

(ii) We prove an equivalent proposition "$\text{Res}(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}'_1 \end{smallmatrix}\right)}) \cap \text{Res}(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_2 \\ \boldsymbol{M}'_2 \end{smallmatrix}\right)}) = \text{Res}(\boldsymbol{f}_{\boldsymbol{M}_1}) \cap \text{Res}(\boldsymbol{f}_{\boldsymbol{M}_2}) \Leftarrow \text{rank}(\left(\begin{smallmatrix} \boldsymbol{M}_1 & \boldsymbol{M}_2 \\ \boldsymbol{M}'_1 & \boldsymbol{M}'_2 \end{smallmatrix}\right)) = \text{rank}(\left(\begin{smallmatrix} \boldsymbol{M}_1 & \boldsymbol{M}_2 \end{smallmatrix}\right))$". $\text{rank}(\left(\begin{smallmatrix} \boldsymbol{M}_1 & \boldsymbol{M}_2 \\ \boldsymbol{M}'_1 & \boldsymbol{M}'_2 \end{smallmatrix}\right)) = \text{rank}(\left(\begin{smallmatrix} \boldsymbol{M}_1 & \boldsymbol{M}_2 \end{smallmatrix}\right))$ means that any row in $\left(\begin{smallmatrix} \boldsymbol{M}'_1 & \boldsymbol{M}'_2 \end{smallmatrix}\right)$ is linearly dependent to $\left(\begin{smallmatrix} \boldsymbol{M}_1 & \boldsymbol{M}_2 \end{smallmatrix}\right)$. Therefore, there exists $\boldsymbol{L} \in \mathbb{R}^{s' \times s}$ so that $\left(\begin{smallmatrix} \boldsymbol{M}_1 & \boldsymbol{M}_2 \\ \boldsymbol{M}'_1 & \boldsymbol{M}'_2 \end{smallmatrix}\right) = \left(\begin{smallmatrix} \boldsymbol{I} \\ \boldsymbol{L} \end{smallmatrix}\right)\left(\begin{smallmatrix} \boldsymbol{M}_1 & \boldsymbol{M}_2 \end{smallmatrix}\right)$. Correspondingly, $\left(\begin{smallmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}'_1 \end{smallmatrix}\right) = \left(\begin{smallmatrix} \boldsymbol{I} \\ \boldsymbol{L} \end{smallmatrix}\right)\boldsymbol{M}_1$ and $\left(\begin{smallmatrix} \boldsymbol{M}_2 \\ \boldsymbol{M}'_2 \end{smallmatrix}\right) = \left(\begin{smallmatrix} \boldsymbol{I} \\ \boldsymbol{L} \end{smallmatrix}\right)\boldsymbol{M}_2$. For any $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ with $\boldsymbol{M}_1\boldsymbol{P}_\pi\boldsymbol{x}_{1\pi} = \boldsymbol{M}_2\boldsymbol{P}_\pi\boldsymbol{x}_{2\pi}$, $\left(\begin{smallmatrix} \boldsymbol{I} \\ \boldsymbol{L} \end{smallmatrix}\right)\boldsymbol{M}_1\boldsymbol{P}_\pi\boldsymbol{x}_{1\pi} = \left(\begin{smallmatrix} \boldsymbol{I} \\ \boldsymbol{L} \end{smallmatrix}\right)\boldsymbol{M}_2\boldsymbol{P}_\pi\boldsymbol{x}_{2\pi}$, and therefore $\left(\begin{smallmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}'_1 \end{smallmatrix}\right)\boldsymbol{P}_\pi\boldsymbol{x}_{1\pi} = \left(\begin{smallmatrix} \boldsymbol{M}_2 \\ \boldsymbol{M}'_2 \end{smallmatrix}\right)\boldsymbol{P}_\pi\boldsymbol{x}_{2\pi}$, where π is the ordering of input elements. Thus for any $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, $\boldsymbol{f}_{\boldsymbol{M}_1}(\boldsymbol{x}_1) = \boldsymbol{f}_{\boldsymbol{M}_2}(\boldsymbol{x}_2) \in \text{Res}(\boldsymbol{f}_{\boldsymbol{M}_1}) \cap \text{Res}(\boldsymbol{f}_{\boldsymbol{M}_2}) \Rightarrow \boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}'_1 \end{smallmatrix}\right)}(\boldsymbol{x}_1) = \boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_2 \\ \boldsymbol{M}'_2 \end{smallmatrix}\right)}(\boldsymbol{x}_2) \in \text{Res}(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}'_1 \end{smallmatrix}\right)}) \cap \text{Res}(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_2 \\ \boldsymbol{M}'_2 \end{smallmatrix}\right)})$. Hence, $\text{Res}(\boldsymbol{f}_{\boldsymbol{M}_1}) \cap \text{Res}(\boldsymbol{f}_{\boldsymbol{M}_2}) \subseteq \text{Res}(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}'_1 \end{smallmatrix}\right)}) \cap \text{Res}(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_2 \\ \boldsymbol{M}'_2 \end{smallmatrix}\right)})$. According to Proposition 2(i), $\text{Res}(\boldsymbol{f}_{\boldsymbol{M}_1}) \cap \text{Res}(\boldsymbol{f}_{\boldsymbol{M}_2}) = \text{Res}(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}'_1 \end{smallmatrix}\right)}) \cap \text{Res}(\boldsymbol{f}_{\left(\begin{smallmatrix} \boldsymbol{M}_2 \\ \boldsymbol{M}'_2 \end{smallmatrix}\right)})$. □

E PROOF OF PROPOSITION 3

Proof. Since $\boldsymbol{M}_1 \in \mathbb{R}^{s\times n_1}$, $\boldsymbol{M}_2 \in \mathbb{R}^{s\times n_2}$ and $\text{rank}(\begin{pmatrix} \boldsymbol{M}_1 & \boldsymbol{M}_2 \end{pmatrix}) = n_1 + n_2$, we have $\text{rank}(\boldsymbol{M}_1) = n_1$ and $\text{rank}(\boldsymbol{M}_2) = n_2$. According to Proposition 1, $\boldsymbol{f}_{\boldsymbol{M}_1}$ and $\boldsymbol{f}_{\boldsymbol{M}_2}$ are injective.

We build the system of linear equations $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, where $\boldsymbol{x} \in \mathbb{R}^{n_1+n_2}$, and $\boldsymbol{A} = \begin{pmatrix} \boldsymbol{M}_1 & \boldsymbol{M}_2 \end{pmatrix}\begin{pmatrix} \boldsymbol{I}^{n_1} & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{I}^{n_2} \end{pmatrix} \in \mathbb{R}^{s\times(n_1+n_2)}$. Then, $\text{rank}(\boldsymbol{A}) = \text{rank}(\begin{pmatrix} \boldsymbol{M}_1 & \boldsymbol{M}_2 \end{pmatrix}\begin{pmatrix} \boldsymbol{I}^{n_1} & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{I}^{n_2} \end{pmatrix}) = \text{rank}(\begin{pmatrix} \boldsymbol{M}_1 & \boldsymbol{M}_2 \end{pmatrix}) = n_1 + n_2$, which means $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ has no non-zero solutions. Let $\boldsymbol{x}' = (\boldsymbol{x}[1], \boldsymbol{x}[2], \cdots, \boldsymbol{x}[n_1])$ and $\boldsymbol{x}'' = (\boldsymbol{x}[n_1+1], \boldsymbol{x}[n_1+2], \cdots, \boldsymbol{x}[n_1+n_2])$ such that $\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}' \\ \boldsymbol{x}'' \end{pmatrix}$. For any $\boldsymbol{x} \neq \boldsymbol{0}$,

$$\boldsymbol{A}\boldsymbol{x} = (\boldsymbol{M}_1 \quad \boldsymbol{M}_2)\begin{pmatrix} \boldsymbol{I}^{n_1} & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{I}^{n_2} \end{pmatrix}\begin{pmatrix} \boldsymbol{x}' \\ \boldsymbol{x}'' \end{pmatrix} = \boldsymbol{M}_1\boldsymbol{x}' - \boldsymbol{M}_2\boldsymbol{x}'' \neq \boldsymbol{0}.$$

Therefore, for any $\boldsymbol{x}' \in \mathbb{R}^{n_1}$, $\boldsymbol{x}'' \in \mathbb{R}^{n_2}$ and $\boldsymbol{x}', \boldsymbol{x}'' \neq \boldsymbol{0}$, $\boldsymbol{M}_1\boldsymbol{x}' \neq \boldsymbol{M}_2\boldsymbol{x}''$, hence $\boldsymbol{M}_1(\boldsymbol{P}_\pi\boldsymbol{x}'_\pi) \neq \boldsymbol{M}_2(\boldsymbol{P}_\pi\boldsymbol{x}''_\pi)$ for any $\boldsymbol{P}_\pi\boldsymbol{x}'_\pi$ and $\boldsymbol{P}_\pi\boldsymbol{x}''_\pi$. As a result, $\text{Res}(\boldsymbol{f}_{\boldsymbol{M}_1}) \cap \text{Res}(\boldsymbol{f}_{\boldsymbol{M}_2}) = \emptyset$. □

F PROOF OF PROPOSITION 4

Proof. For Multi-head GAT, there are two types of implementations on aggregating each head, $concatenation$ and $average$. Here, we only consider the average aggregation implementation.

$$\begin{aligned}
\boldsymbol{h}_v^{(t)} &= \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_v}\alpha_{vu}^k\boldsymbol{W}^k\boldsymbol{h}_u^{(t-1)}\right) \\
&= \sigma\left(\frac{1}{K}\sum_{j\in\mathcal{N}_v}\sum_{k=1}^{K}\boldsymbol{W}^k(\alpha_{vu}^k\boldsymbol{h}_u^{(t-1)})\right) \\
&= \sigma\left(\frac{1}{K}\sum_{j\in\mathcal{N}_v}(||_{k=1}^{K}\boldsymbol{W}^k)(||_{k=1}^{K}\alpha_{vu}^k\boldsymbol{h}_u^{(t-1)})\right) \\
&= \sigma\left(\frac{1}{K}\sum_{j\in\mathcal{N}_v}\boldsymbol{W}\text{vec}(\boldsymbol{\alpha}_{vu}\boldsymbol{h}_u^{(t-1)T})\right) \\
&= \sigma\left(\frac{1}{K}\boldsymbol{W}\sum_{j\in\mathcal{N}_v}\text{vec}(\boldsymbol{\alpha}_{vu}\boldsymbol{h}_u^{(t-1)T})\right),
\end{aligned}$$

where $\boldsymbol{W}^k \in \mathbb{R}^{d\times d}$ is the trainable matrix for the k-th head, and $\boldsymbol{W} = ||_{k=1}^{K}\boldsymbol{W}^k \in \mathbb{R}^{kd\times d}$ is the concatenation of the trainable matrix in all K heads;

$$\boldsymbol{\alpha}_{vu} = \text{softmax}(\text{LeakyReLU}\begin{pmatrix} \boldsymbol{a}^{1T}[\boldsymbol{W}^1\boldsymbol{h}_v^{(t-1)}||\boldsymbol{W}^1\boldsymbol{h}_u^{(t-1)}] \\ \boldsymbol{a}^{2T}[\boldsymbol{W}^2\boldsymbol{h}_v^{(t-1)}||\boldsymbol{W}^2\boldsymbol{h}_u^{(t-1)}] \\ \vdots \\ \boldsymbol{a}^{KT}[\boldsymbol{W}^1\boldsymbol{h}_v^{(t-1)}||\boldsymbol{W}^K\boldsymbol{h}_u^{(t-1)}] \end{pmatrix}).$$

Let $\tilde{\boldsymbol{a}}^\star = (\boldsymbol{a}_1^\star, \boldsymbol{a}_2^\star, \cdots, \boldsymbol{a}_K^\star)$ and $\tilde{\boldsymbol{a}}'^\star = (\boldsymbol{a}_{K+1}^\star, \boldsymbol{a}_{K+2}^\star, \cdots, \boldsymbol{a}_{2K}^\star)$. Then,

$$
\begin{aligned}
&\begin{pmatrix} \boldsymbol{a}^{1T}[\boldsymbol{W}^1\boldsymbol{h}_v^{(t-1)}||\boldsymbol{W}^1\boldsymbol{h}_u^{(t-1)}] \\ \boldsymbol{a}^{2T}[\boldsymbol{W}^2\boldsymbol{h}_v^{(t-1)}||\boldsymbol{W}^2\boldsymbol{h}_u^{(t-1)}] \\ \vdots \\ \boldsymbol{a}^{KT}[\boldsymbol{W}^1\boldsymbol{h}_v^{(t-1)}||\boldsymbol{W}^K\boldsymbol{h}_u^{(t-1)}] \end{pmatrix} \\
&= \begin{pmatrix} [\tilde{\boldsymbol{a}}^1||\tilde{\boldsymbol{a}}'^1]^T[\boldsymbol{W}^1\boldsymbol{h}_v^{(t-1)}||\boldsymbol{W}^1\boldsymbol{h}_u^{(t-1)}] \\ [\tilde{\boldsymbol{a}}^2||\tilde{\boldsymbol{a}}'^2]^T[\boldsymbol{W}^2\boldsymbol{h}_v^{(t-1)}||\boldsymbol{W}^2\boldsymbol{h}_u^{(t-1)}] \\ \vdots \\ [\tilde{\boldsymbol{a}}^K||\tilde{\boldsymbol{a}}'^K]^T[\boldsymbol{W}^1\boldsymbol{h}_v^{(t-1)}||\boldsymbol{W}^K\boldsymbol{h}_u^{(t-1)}] \end{pmatrix} \\
&= \begin{pmatrix} \tilde{\boldsymbol{a}}^{1T}\boldsymbol{W}^1\boldsymbol{h}_v^{(t-1)} + \tilde{\boldsymbol{a}}'^{1T}\boldsymbol{W}^1\boldsymbol{h}_u^{(t-1)} \\ \tilde{\boldsymbol{a}}^{2T}\boldsymbol{W}^2\boldsymbol{h}_v^{(t-1)} + \tilde{\boldsymbol{a}}'^{2T}\boldsymbol{W}^2\boldsymbol{h}_u^{(t-1)} \\ \vdots \\ \tilde{\boldsymbol{a}}^{KT}\boldsymbol{W}^K\boldsymbol{h}_v^{(t-1)} + \tilde{\boldsymbol{a}}'^{KT}\boldsymbol{W}^K\boldsymbol{h}_u^{(t-1)} \end{pmatrix} \\
&= \begin{pmatrix} \tilde{\boldsymbol{a}}^{1T}\boldsymbol{W}^1\boldsymbol{h}_v^{(t-1)} \\ \tilde{\boldsymbol{a}}^{2T}\boldsymbol{W}^2\boldsymbol{h}_v^{(t-1)} \\ \vdots \\ \tilde{\boldsymbol{a}}^{KT}\boldsymbol{W}^K\boldsymbol{h}_v^{(t-1)} \end{pmatrix} + \begin{pmatrix} \tilde{\boldsymbol{a}}'^{1T}\boldsymbol{W}^1\boldsymbol{h}_u^{(t-1)} \\ \tilde{\boldsymbol{a}}'^{2T}\boldsymbol{W}^2\boldsymbol{h}_u^{(t-1)} \\ \vdots \\ \tilde{\boldsymbol{a}}'^{KT}\boldsymbol{W}^K\boldsymbol{h}_u^{(t-1)} \end{pmatrix} \\
&= \begin{pmatrix} \tilde{\boldsymbol{a}}^{1T} & & & \\ & \tilde{\boldsymbol{a}}^{2T} & & \\ & & \ddots & \\ & & & \tilde{\boldsymbol{a}}^{KT} \end{pmatrix} \begin{pmatrix} \boldsymbol{W}^1 \\ \boldsymbol{W}^2 \\ \vdots \\ \boldsymbol{W}^K \end{pmatrix} \boldsymbol{h}_v^{(t-1)} + \begin{pmatrix} \tilde{\boldsymbol{a}}'^{1T} & & & \\ & \tilde{\boldsymbol{a}}'^{2T} & & \\ & & \ddots & \\ & & & \tilde{\boldsymbol{a}}'^{KT} \end{pmatrix} \begin{pmatrix} \boldsymbol{W}^1 \\ \boldsymbol{W}^2 \\ \vdots \\ \boldsymbol{W}^K \end{pmatrix} \boldsymbol{h}_u^{(t-1)} \\
&= \text{diag}(\tilde{\boldsymbol{a}}^{1T}, \tilde{\boldsymbol{a}}^{2T}, \cdots, \tilde{\boldsymbol{a}}^{KT})\boldsymbol{W}\boldsymbol{h}_v^{(t-1)} + \text{diag}(\tilde{\boldsymbol{a}}'^{1T}, \tilde{\boldsymbol{a}}'^{2T}, \cdots, \tilde{\boldsymbol{a}}'^{KT})\boldsymbol{W}\boldsymbol{h}_u^{(t-1)} \\
&= [\text{diag}(\tilde{\boldsymbol{a}}^{1T}, \tilde{\boldsymbol{a}}^{2T}, \cdots, \tilde{\boldsymbol{a}}^{KT})||\text{diag}(\tilde{\boldsymbol{a}}'^{1T}, \tilde{\boldsymbol{a}}'^{2T}, \cdots, \tilde{\boldsymbol{a}}'^{KT})][\boldsymbol{W}\boldsymbol{h}_v^{(t-1)}||\boldsymbol{W}\boldsymbol{h}_u^{(t-1)}].
\end{aligned}
$$

Therefore, multi-head GAT is an implementation of ExpandingConv as follows:

$$
\begin{cases}
\boldsymbol{\alpha}_{vu} = \text{softmax}(\text{LeakyReLU}([\text{diag}(\tilde{\boldsymbol{a}}^{1T}, \tilde{\boldsymbol{a}}^{2T}, \cdots, \tilde{\boldsymbol{a}}^{KT}) \\
\qquad ||\text{diag}(\tilde{\boldsymbol{a}}'^{1T}, \tilde{\boldsymbol{a}}'^{2T}, \cdots, \tilde{\boldsymbol{a}}'^{KT})][\boldsymbol{W}\boldsymbol{h}_v^{(t-1)}||\boldsymbol{W}\boldsymbol{h}_u^{(t-1)}])) \\
\boldsymbol{h}_v^{(t)} = \sigma\left(\frac{1}{K}\boldsymbol{W}\sum_{j\in\mathcal{N}_v}\text{vec}(\boldsymbol{\alpha}_{vu}\boldsymbol{h}_u^{(t-1)T})\right),
\end{cases}
$$

□

G Comparisons with Multi-aggregator Implementations

ExpandingConv can also be considered as a kind of multi-aggregator scheme. In Equation 5, each row of $\boldsymbol{M}_{v_i}$ can be viewed as a weighted aggregator where the weight coefficients are learned from data. Proposition 1 shows that to obtain higher distinguishing strength by utilizing more aggregators, the weight coefficients of newly added aggregators should be linearly independent to all existing aggregators. The distinguishing strength of weighted aggregators is incomparable with basic aggregators. However, since each row of $\boldsymbol{M}_{v_i}$ is equivalent to an independent aggregator, one can simply modify the implementation of $\boldsymbol{f}_{\text{local}}(u, v)$ to obtain the variant whose distinguishing strength is strict stronger than basic aggregators as follows:

$$
\begin{aligned}
&\text{ExpandingConv}|_{\boldsymbol{m}_{uv}'^{(t)}=[\boldsymbol{m}_{uv}^{(t)}||1]} \succ \text{SUM}, \\
&\text{ExpandingConv}|_{\boldsymbol{m}_{uv}'^{(t)}=[\boldsymbol{m}_{uv}^{(t)}||1||\frac{1}{|\mathcal{N}(v)|}]} \succ \text{SUM} \otimes \text{MEAN}.
\end{aligned}
$$

Compared with lerveraging multiple basic aggregators in Corso et al. (2020) and Dehmamy et al. (2019), lerveraging weighted aggregator allows for variable numbers of aggregators. Meanwhile, the weighted coefficients are learned from data, which can better capture relevant structural patterns.

H DETAILS OF EXPERIMENTAL SETUP

Datasets. Benchmark datasets for graph kernels provided by TU Kersting et al. (2016) suffer from their small scales of data, making them not sufficient to evaluate the performance of models Dwivedi et al. (2020). Our evaluations are conducted on graph property predictions datasets ogbg-ppa, ogbg-code, ogbg-molhiv in OGB Weihua Hu (2020) and QM9 Ramakrishnan et al. (2014); Wu et al. (2018); Ruddigkeit et al. (2012) which are large-scale graph datasets including graph classification and graph regression tasks. ogbg-ppa is extracted from the protein-protein association networks with large and densely connected graphs. ogbg-code is a collection of Abstract Syntax Trees (ASTs) obtained from Python method definitions with large and sparse graphs. ogbg-molhiv is molecular property prediction datasets with relative small graphs. QM9 consists 134K small organic molecules with the task to predict 12 targets for each molecule. All data is obtained from pytorch-geometric library Fey & Lenssen (2019).

Configurations. We use the default dataset splits for OGB. The QM9 dataset is randomly split into 80% train, 10% validation and 10% test as given in Morris et al. (2019); Maron et al. (2019). We use the concatenation of hidden features from all layers to compute the entire graph representations Xu et al. (2018). In our tests, all models are equipped with batch normalization Ioffe & Szegedy (2015) on each hidden layer when evaluating on OGB, and are not when evaluating on QM9.

Table 3: Hyperparameter settings for OGB.

	ogbg-ppa		ogbg-molhiv		ogbg-code	
	ExpC*-1, ExpC-s	CombC*, CombC	ExpC*-1, ExpC-s	CombC*, CombC	ExpC*-1, ExpC-s	CombC*, CombC
batch size	32	32	64	64	64	64
layers	4	4	3	3	4	4
hidden	256	256	64	64	512	512
lr	0.0005	0.0002	0.0001	0.0001	0.0001	0.0001
step size	20	20	5	5	5	5
lr decay	0.8	0.7	0.7	0.7	0.6	0.6
dropout	0.5	0.5	0.5	0.5	0.5	0.5
readout	SUM	SUM	MEAN	MEAN	MEAN	MEAN

The shared hyperparameter settings of ExpC*-1, ExpC-s, CombC* and CombC on all 12 targets of QM9: batch sizes = 64; lr = 0.0001; step size = 30; lr decay = 0.85; readout = SUM. hidden = 256 for ExpC*-1 and ExpC-s; hidden = 512 for CombC* and CombC. Table 4 gives the individual hyperparameter settings of each model on each target, including the number of layers.

Table 4: Number of layers for QM9.

	μ	α	ϵ_{homo}	ϵ_{lumo}	$\Delta\epsilon$	$\langle R^2 \rangle$	$ZPVE$	U_0	U	H	G	C_v
ExpC*-1,ExpC-s	5	4	5	4	4	4	4	5	4	4	4	4
CombC*,CombC	5	4	5	4	4	5	4	5	4	4	4	4

I MORE EXPERIMENTAL RESULTS

We present more results of ablation studies on OGB and QM9, which demonstrate the effectiveness of ExpandingConv, CombConv and Re-SUM.

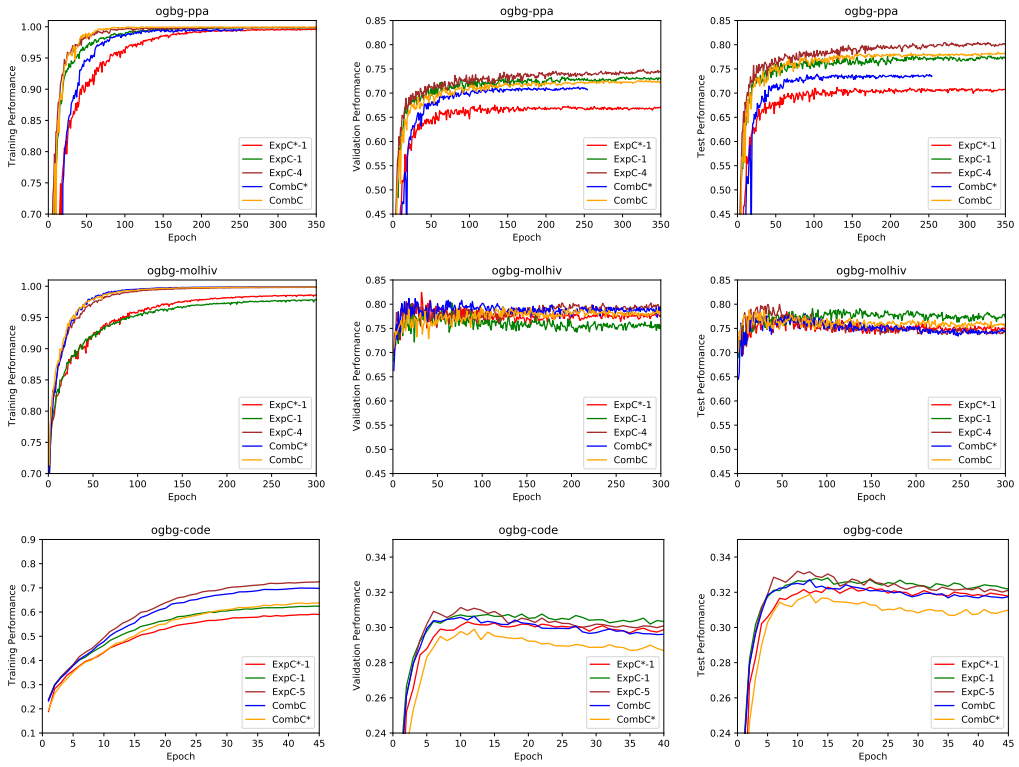

Figure 1: Learning curves on ogbg-ppa, ogbg-molhiv ogbg-code.

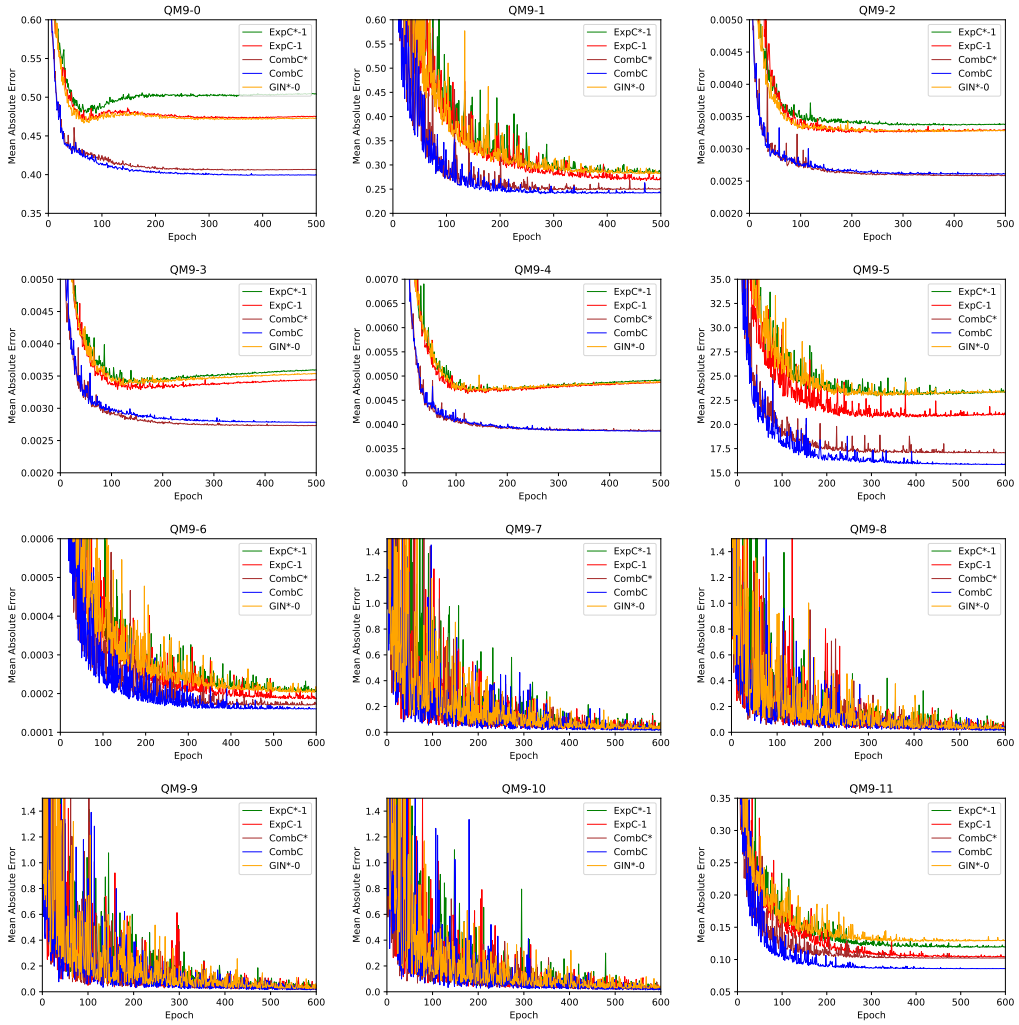

Figure 2: Effectiveness of Re-SUM on QM9.

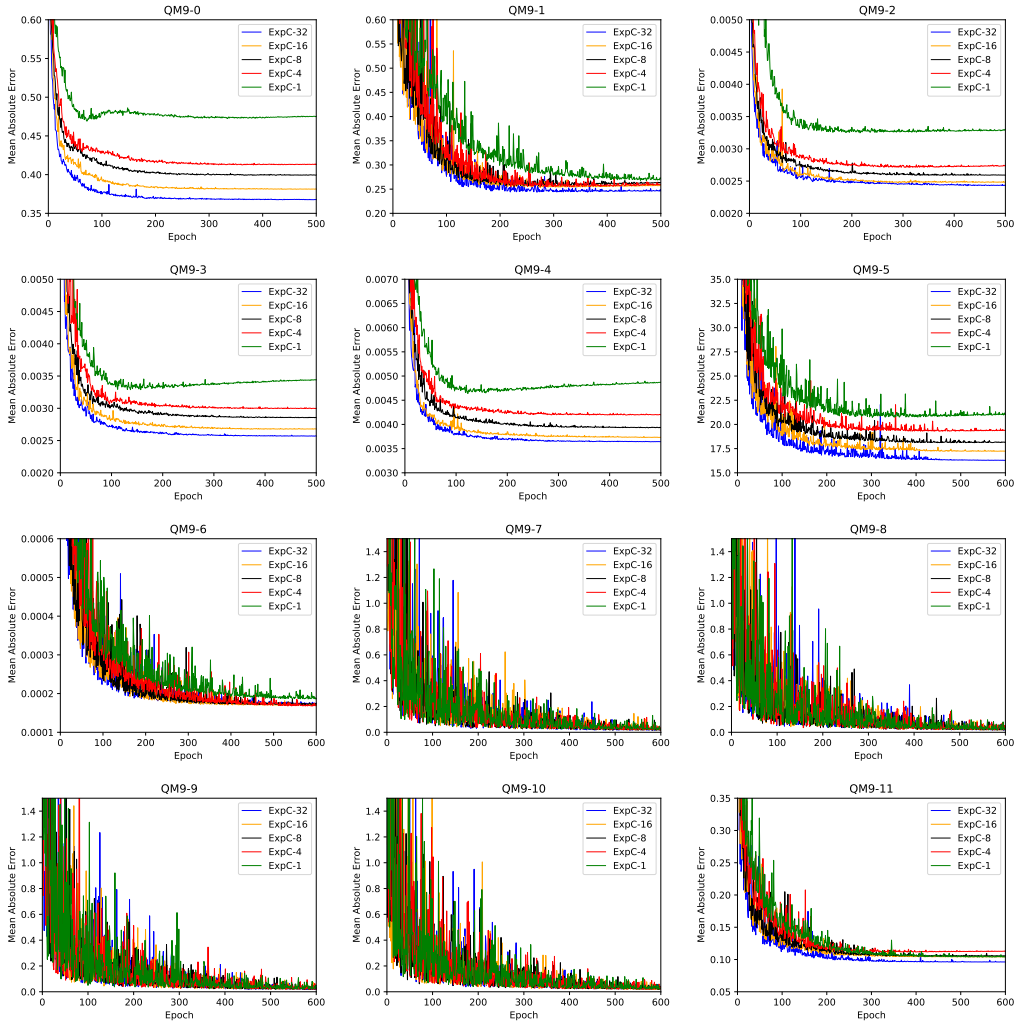

Figure 3: Effectiveness of powerful aggregators on QM9.