Are Graph Transformers Necessary? Efficient Long-Range Message Passing with Fractal Nodes in MPNNs

Jeongwhan Choi[1], **Seungjun Park**[1], **Sumin Park**[1], **Sung-Bae Cho**[2], **Noseong Park**[1]
[1]Korea Advanced Institute of Science & Technology
[2]Yonsei University
{jeongwhan.choi, tmdwns1127, psmiz, noseong}@kaist.ac.kr, sbcho@yonsei.ac.kr

Abstract

Graph Neural Networks (GNNs) have emerged as powerful tools for learning on graph-structured data, but often struggle to balance local and global information. While graph Transformers aim to address this by enabling long-range interactions, they often overlook the inherent locality and efficiency of Message Passing Neural Networks (MPNNs). We propose a new concept called '*fractal nodes*', inspired by the fractal structure observed in real-world networks. Our approach is based on the intuition that graph partitioning naturally induces fractal structure, where subgraphs often reflect the connectivity patterns of the full graph. Fractal nodes are designed to coexist with the original nodes and adaptively aggregate subgraph-level feature representations, thereby enforcing feature similarity within each subgraph. We show that fractal nodes alleviate the over-squashing problem by providing direct shortcut connections that enable long-range propagation of subgraph-level representations. Experiment results show that our method improves the expressive power of MPNNs and achieves comparable or better performance to graph Transformers while maintaining the computational efficiency of MPNN by improving the long-range dependencies of MPNN.

Code — https://github.com/jeongwhanchoi/MPNN-FN
Extended version — https://www.arxiv.org/abs/2511.13010

1 Introduction

GNNs have become powerful tools for learning from graph-structured data across several domains (Kipf and Welling 2017; Veličković et al. 2018; Choi, Jeon, and Park 2021; Choi et al. 2023; Jeong et al. 2025). At the core of this field are MPNN (Gilmer et al. 2017), which propagates information by iteratively exchanging messages between neighboring nodes. However, MPNNs face limitations, particularly over-smoothing (Nt and Maehara 2019) and over-squashing (Alon and Yahav 2021). To address these challenges, Transformer architectures (Vaswani et al. 2017) have been adapted for graph learning, applying self-attention mechanisms to enable long-range interactions by treating all nodes as tokens (Dwivedi and Bresson 2021; Wu et al. 2021; Choi et al. 2024b). While graph Transformers effectively capture global context, they often overlook the inherent locality of MPNNs (Xing et al. 2024). Although approaches such as GraphGPS (Rampášek et al. 2022) attempt to combine MPNN and Transformer node representations to balance local and global information, the computational complexity of Transformers remains a challenge.

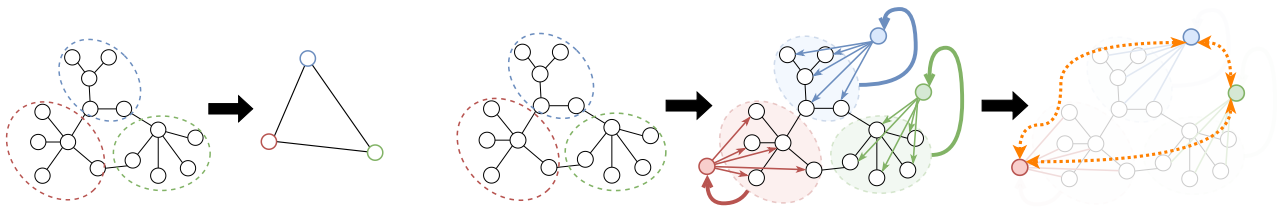
Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Motivation. The limitations of both MPNN and graph Transformers motivate us to seek a novel approach that balances local and global information while preserving scalability. Our inspiration comes from the concept of fractals (Mandelbrot 1983), which describes systems where similar patterns recur across scales. Many real-world graphs show this fractal structure, where structural patterns repeat across different parts of the graph (Dill et al. 2002; Kim and Kahng 2010; Chen et al. 2020). Fractal structures in graphs are commonly studied through renormalization (Song, Havlin, and Makse 2005) (see Fig. 1(a)), where groups of nodes are coarsened into "super-nodes" to analyze global properties emerging from local structures. In contrast to renormalization, one may consider partitioning the graph into subgraphs and learning representative features for each subgraph — without changing the original graph topology. This alternative perspective leads us to the following question:

Can fractal-inspired representations enhance long-range message passing, while preserving the efficiency of MPNNs instead of graph Transformers?

Our answer is "yes", and we introduce our main idea: *fractal nodes* for enforcing feature similarity.

Main idea: graph partitioning causes *fractal structure*, and each *fractal node* enforces feature similarity in each partition. We propose a novel concept called '*fractal nodes*', inspired by the fractal structure observed in real-world networks. We build on the view that graph partitioning naturally causes fractal structure, where subgraphs reflect the connectivity patterns of the full graph. Rather than replacing groups of nodes with super nodes as in renormalization, we retain the original graph and introduce fractal nodes (🔴, 🔵, 🟢) that coexist with the original nodes (see Fig. 1(b)). Each fractal node serves as a representative of its subgraph and is connected to all nodes in its subgraph. These nodes aggregate subgraph-level information and enforce *feature similarity* among the nodes within the same subgraph. To construct these representations, each fractal node adaptively combines both the low-frequency (global) and high-frequency (local)



(a) Renormalized graph

(b) Graph where our fractal nodes are connected

Figure 1: Conceptual comparison of renormalization and our fractal node process. (a) In renormalization, the original graph is replaced by a single node according to each box-covering method, creating a coarsened network. (b) After partitioning the graph, *fractal nodes* (🔴, 🔵, 🟢) are created for each subgraph, which represent the subgraph nodes while having similar features at different scales. Then, we propagate the information to the original nodes (our proposed FN). We also support the long-distance interactions (orange dashed lines) between fractal nodes (our proposed FN_M).

components of the features in its subgraph. This goes beyond simple mean pooling, which only retains the lowest-frequency signal. These one-hop shortcut connections between the fractal nodes and their subgraph nodes effectively reduce the information propagation distance. This theoretical reduction in effective resistance between nodes empirically supports the mitigation of over-squashing (Alon and Yahav 2021) and improves signal propagation. Furthermore, we apply an MLP-Mixer (Tolstikhin et al. 2021) at the final layer to flexibly mix the representations of the fractal nodes. This allows long-range interactions between subgraphs without relying on multi-hop message passing, which leads to signal degradation in standard MPNNs.

Contributions. We propose a novel paradigm, *fractal nodes*, which improves message passing in GNNs by enforcing feature similarity within subgraphs, using the fractal structure caused by graph partitioning. Our main contributions are as follows:

- We introduce *fractal nodes*, a plug-in component that captures subgraph-level information, and integrate them into MPNNs (Sec. 3). We discuss the properties of *fractal nodes* (Sec. 4).
- We analyze the *fractal nodes* in reducing effective resistance and demonstrate that they mitigate the over-squashing problem (Sec. 5.1). We also show improved expressive power compared to standard MPNNs (Sec. 5.2).
- Extensive experiments on various benchmark datasets show that MPNNs augmented with fractal nodes achieve performance comparable to or better than state-of-the-art graph Transformer-based models (Sec. 5.3), while improving the scalability and maintaining efficiency of standard MPNNs (Sec. 5.4).

2 Preliminaries & Related Work

We first introduce MPNNs and their limitations, then discuss related work. Due to space constraints, we provide additional related work in Appendix (Choi et al. 2025).

Message passing neural network. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we use $\mathcal{V}$ and $\mathcal{E}$ to denote its nodes and edges, respectively. The nodes are indexed by v and u such that $v, u \in \mathcal{V}$, and an edge connecting the nodes v and u is denoted by $(v, u) \in \mathcal{E}$. We consider the case where each node v has a hidden vector $h_v^{(\ell)} \in \mathbb{R}^{\dim(h)}$, where $\dim(h)$ is the size of the hidden dimension, and ℓ is the number of layers. MPNNs iteratively update node representations:

$$h_v^{(\ell+1)} = \varphi(h_v^{(\ell)}, \psi^{(\ell)}(\{h_u^{(\ell)} : u \in \mathcal{N}(v)\})), \tag{1}$$

where $\psi^{(\ell)}$ and $\varphi^{(\ell)}$ are the aggregation function and the update function.

Limitations of MPNNs. MPNNs have been investigated for their *expressive power* limitations and *over-squashing* problems. Simple MPNNs are known to be only as powerful as the 1-Weisfeler-Leman graph isomorphism test (Xu et al. 2019). The over-squashing problem occurs when MPNNs struggle to propagate information along long paths, resulting in loss of information when aggregating from too many neighbors into a fixed-sized node feature vector (Alon and Yahav 2021; Di Giovanni et al. 2023). This over-squashing phenomenon can be precisely characterized through effective resistance (Black et al. 2023). The effective resistance between nodes u and v measures the difficulty of information flow between them:

$$R(u, v) = (\mathbf{1}_u - \mathbf{1}_v)^\top \mathbf{L}^+ (\mathbf{1}_u - \mathbf{1}_v), \tag{2}$$

where $\mathbf{L}^+$ is the pseudoinverse of the graph Laplacian and $\mathbf{1}_u$ is the indicator vector for node u. Intuitively, higher resistance indicates fewer paths between nodes, creating bottlenecks where information from many neighbors must be compressed into fixed-sized vectors. This quantitative framework reveals why local message passing cannot effectively capture both local and global context, leading to the development of graph Transformers, which use self-attention to enable "everything is connected to everything". We use this effective resistance in Sec. 4 to analyze how fractal nodes alleviate over-squashing while maintaining the efficiency of MPNNs.

Augmented MPNNs. To improve information flow and address the limitations of MPNNs, various strategies have been proposed (Di Giovanni et al. 2023). One approach is incorporating additional global graph features during representation learning (Gilmer et al. 2017; Hu et al. 2020). The expanded

width-aware message passing (Choi et al. 2024a) offers an alternative that mitigates over-squashing without rewiring by selectively expanding the width of high-centrality nodes. Another effective method is rewiring the input graph to enhance connectivity and alleviate structural bottlenecks (Gasteiger, Weißenberger, and Günnemann 2019; Black et al. 2023; Karhadkar, Banerjee, and Montufar 2023; Yu et al. 2025). These adjustments allow for more effective information flow within the network. Another example of graph augmentation is a virtual node. This heuristic, introduced by Gilmer et al. (2017), has been observed to improve performance on various tasks. Further analyses by Cai et al. (2023); Southern et al. (2025) have explored the role of virtual nodes in mitigating under-reaching and over-smoothing.

Graph Transformers. Because of the success of Transformers, previous works have adapted this architecture for graph learning (Dwivedi and Bresson 2021). Dwivedi and Bresson (2021) proposed using graph Laplacian eigenvectors as node positional encodings. GraphGPS (Rampášek et al. 2022) provides a general framework combining MPNNs with Transformer components, while Graphormer(Ying et al. 2021) uses attention mechanisms to estimate several types of encodings, including centrality, spatial, and edge encodings. Wu et al. (2021) applies the MPNN directly to all nodes and then applies a Transformer, which is computationally intensive. He et al. (2023) generalize ViT (Dosovitskiy et al. 2021) to graphs and Ma et al. (2023) show that adding inductive biases to graph Transformers removes the need for MPNN modules in GraphGPS. While graph Transformers effectively capture long-range dependencies, they typically scale quadratically with the number of nodes $\mathcal{O}(|\mathcal{N}|^2)$ compared to the linear scaling of MPNNs $\mathcal{O}(|\mathcal{E}|)$, motivating our work to develop more efficient architectures that preserve global information capture.

3 Message Passing with Fractal Nodes

In this section, we propose fractal nodes and explain how they contribute to overcoming the limitations of existing MPNNs. We describe how to create fractal nodes, how to augment the MPNNs, and how to implement interactions between intra and inter-subgraphs.

Notation. Let $\{\mathcal{V}_1, \ldots, \mathcal{V}_C\}$ be the set of node subsets corresponding to C subgraphs, where C is the number of subgraphs. $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c)$ is the induced subgraph of $\mathcal{G}$. We define $h_{v,c}^{(\ell)}$ as the hidden vector of node v of the c-th subgraph in layer ℓ, and $f_c^{(\ell)}$ as the hidden vector of the fractal node of the c-th subgraph in the ℓ-th layer.

Message passing with *fractal nodes*. We first introduce the message passing process, including fractal nodes. The message passing process proceeds as follows:

$$\widetilde{h}_{v,c}^{(\ell+1)} = \varphi^{(\ell)}(h_{v,c}^{(\ell)}, \psi^{(\ell)}(h_{u,c}^{(\ell)} : u \in \mathcal{N}_v)), \tag{3}$$

$$f_c^{(\ell+1)} = \varphi_{\mathsf{FN}}^{(\ell)}(f_c^{(\ell)}, \psi_{\mathsf{FN}}^{(\ell)}(\widetilde{h}_{u,c}^{(\ell+1)} : u \in \mathcal{N}_v)), \tag{4}$$

$$h_{v,c}^{(\ell+1)} = \widetilde{\varphi}^{(\ell)}(\widetilde{h}_{v,c}^{(\ell+1)}, f_c^{(\ell+1)}), \tag{5}$$

where $\mathcal{N}_v$ is the set of neighbors of node v. Eq. (3) performs standard message passing at the node level. Eq. (4) updates the fractal node representations. It aggregates hidden vectors from all nodes in the subgraph, $\mathcal{G}_c$, using the $\widetilde{h}_{u,c}^{(\ell+1)}$, and then updates the fractal nodes. $\psi_{\mathsf{FN}}^{(\ell)}$ and $\varphi_{\mathsf{FN}}^{(\ell)}$ are aggregate and update functions for fractal nodes, which will be explained in more detail. The update function $\widetilde{\varphi}^{(\ell)}$ is the step where the message $f_c^{(\ell+1)}$ is propagated to $h_{v,c}^{(\ell)}$.

How to create *fractal nodes*. As shown in Fig. 1(b), fractal nodes are created from subgraphs. To partition into subgraphs, we consider the METIS (Karypis and Kumar 1998) algorithm for its scalability and efficiency. How we use METIS is provided in Appendix (Choi et al. 2025). In our design, each fractal node plays two roles: (i) structurally, it originates from a subgraph that reflects the recurring connectivity patterns of the original graph, and (ii) functionally, it encodes a subgraph-level feature representation that enforces *feature similarity* among the nodes within the same subgraph. While graph partitioning naturally preserves structural characteristics, our method focuses on learning subgraph-level representations by adaptively combining low-frequency (global) and high-frequency (local) components of node features within each subgraph. To motivate this, we first show that simple mean pooling captures only the direct current (DC) component of the signal.

Theorem 3.1 (Mean pooling as a low-pass filter)**.** *The mean pooling operation applied to the node features is equivalent to extracting the lowest frequency component in the Fourier domain, acting as a low-pass filter that discards all high-frequency information.*

The mean pooling corresponds to extracting the lowest frequency component — also known as the DC component — in the Fourier domain. This DC component captures the global characteristic of the subgraph, but it ignores higher-frequency variations that represent local details. A formal proof is provided in Appendix (Choi et al. 2025).

While Theorem 3.1 shows that mean pooling only captures the DC component, fractal nodes go beyond this limitation by using LPF and HPF. We adaptively rescale the high-frequency component, and combine LPF and HPF together to form fractal nodes:

$$f_c^{(\ell+1)} = \mathsf{LPF}(\{h_{v,c}^{(\ell+1)}\}_{v\in\mathcal{V}_c}) + \omega_c^{(\ell)} \cdot \mathsf{HPF}(\{h_{v,c}^{(\ell+1)}\}_{v\in\mathcal{V}_c}),$$

where $\omega_c^{(\ell)}$ is a learnable parameter controlling the contribution of high-frequency components. We use a learnable scalar, $\omega_c^{(\ell)} \in \mathbb{R}$, or a learnable vector parameter, $\boldsymbol{\omega}_c^{(\ell)} \in \mathbb{R}^{\dim(h)}$. The LPF is computed by averaging the node features within the subgraph, so it can capture global information:

$$\mathsf{LPF}(\{h_{v,c}^{(\ell+1)}\}_{v\in\mathcal{V}_c}) := \frac{1}{|\mathcal{V}_c|}\sum_{v\in\mathcal{V}_c} h_{v,c}^{(\ell+1)},$$

which is analogous to mean pooling and represents the global, low-frequency component of the subgraph. To capture the local details, the HPF is applied by subtracting the low-pass filtered output from the original node hidden vector. This allows the model to retain the local variations:

$$\mathsf{HPF}(\{h_{v,c}^{(\ell+1)}\}_{v\in\mathcal{V}_c}) := h_{v,c}^{(\ell+1)} - \mathsf{LPF}(\{h_{u,c}^{(\ell+1)}\}_{u\in\mathcal{V}_c}).$$

Fractal nodes mixing with MLP-Mixer. We also allow fractal nodes to exchange messages, as the coarsened network in Fig. 1(a) takes advantage of long-distance interactions. To do this, we can apply the MLP-Mixer layer (Tolstikhin et al. 2021) to the fractal nodes in the last layer. This means that we do not need to create a coarsened network, and the MLP-Mixer flexibly mixes the representations of fractal nodes:

$$\widetilde{F} = \mathsf{MLPMixer}(F^{(L)}),\ F^{(L)} = [f_1^{(L)}, \ldots, f_C^{(L)}], \quad (6)$$

where $F^{(L)}$ is the matrix of all fractal node representations at the final layer L. The MLP-Mixer layer consists of token-mixing and channel-mixing steps:

$$U = F^{(L)} + (W_2\rho(W_1\mathsf{LayerNorm}(F^{(L)}))) \quad (7)$$

$$\widetilde{F}^{(L)} = U + (W_4\rho(W_3\mathsf{LayerNorm}(U^T)^T), \quad (8)$$

where ρ is a GELU nonlinearity, and matrices W_1, W_2, W_3, W_4 are learnable weight matrices.

Instance of our framework. We present two variants of our method: FN and FN_M ('Fractal Nodes with Mixer'). The key difference is that FN applies fractal nodes at every layer through the update equations in Eqs. (3) to (5), while FN_M additionally employs an MLP-Mixer at the final layer (see Eq. (6)) to enable direct communication between fractal nodes across different subgraphs. This allows FN_M to capture long-range inter-subgraph dependencies more effectively. We provide instantiations with GCN, GINE, and GatedGCN in Appendix (Choi et al. 2025).

The output layer. Once the final representation h_G is derived, we use a multi-layer perceptron (MLP) as an output layer to predict graph-level outputs:

$$h_G = \mathsf{MeanPool}(H^{(L)} \text{ for FN},\ \widetilde{F}^{(L)} \text{ for } \mathsf{FN}_M),$$
$$y_G = \mathsf{MLP}(h_G),$$

where y_G is either a scalar for regression tasks or a vector for classification tasks, and $H^{(L)} = [h_1^{(L)}, ..., h_{|V|}^{(L)}]$ is the matrix of node representations at the final layer L for all nodes in the graph. For details on our method for node classification tasks, please refer to Appendix (Choi et al. 2025).

4 Properties of Fractal Nodes

In this section, we analyze why fractal nodes are effective and what properties they have, discuss the model complexity, and compare them with previous work.

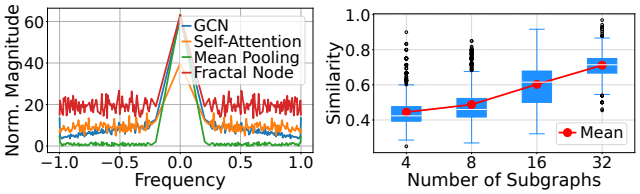
4.1 Why Fractal Nodes Improve Message Passing

Theoretical analyses. We provide theoretical analyses showing that fractal nodes help mitigate over-squashing by reducing the effective resistance between nodes. Full theoretical analyses are provided in Appendix (Choi et al. 2025).

Theorem 4.1 (Resistance reduction)**.** *Let $\mathcal{G}$ be the original graph and $\mathcal{G}_f$ be the augmented graph with fractal nodes. For any nodes $u, v \in \mathcal{G}$, the effective resistance in $\mathcal{G}_f$ satisfies:*

$$R_f(u, v) \leq R(u, v),$$

where $R_f(u, v)$ is the effective resistance in $\mathcal{G}_f$ and $R(u, v)$ is the original effective resistance in $\mathcal{G}$.



(a) (b)

Figure 2: (a) Normalized frequency response on PEPTIDES-STRUCT. (b) Structural similarity of node centrality distribution in PEPTIDES-STRUCT.

This reduction in effective resistance improves signal propagation between distant nodes as follows Theorem 4.2.

Theorem 4.2 (Signal propagation with fractal nodes)**.** *For an MPNN with fractal nodes, the signal propagation between nodes u, v after ℓ layers satisfies:*

$$\|h_u^{(\ell)} - h_v^{(\ell)}\| \leq \exp(-\ell/R_f(u, v))\|h_u^{(0)} - h_v^{(0)}\|,$$

where $R_f(u, v)$ is the effective resistance in $\mathcal{G}_f$.

Since $R_f(u, v) \leq R(u, v)$, fractal nodes improve the worst-case signal propagation bound compared to the original graph. *The proofs and detailed analyses can be found in* Appendix (Choi et al. 2025).

Frequency response analysis. We analyze fractal nodes from a frequency perspective. Fig. 2(a) shows normalized frequency responses. Mean pooling shows a minimal response in the high-frequency domain, which suggests an oversimplification of node representations by losing local details. GCN and self-attention have higher responses than mean pooling in the high-frequency domain, but are still lower than fractal nodes. The improved high-frequency response of fractal nodes preserves fine local details.

Fractal structure and feature similarity. Fractal structure emerges naturally from the graph partitioning process. Since each subgraph is a subset of the original graph, it often reflects the overall connectivity patterns of the full graph. This structural similarity aligns with the concept of fractality, where similar patterns recur across different scales. To quantify this, we compare the betweenness centrality distributions between the original graph and its subgraphs. Betweenness centrality (Freeman 1977) captures both local and global structural importance, particularly in graphs where even low-degree nodes can serve as critical bridges (Kitsak et al. 2007). As shown in Fig. 2(b), the structural similarity increases with the number of subgraphs (see Appendix (Choi et al. 2025) for details). On top of this property, our fractal nodes are designed to enforce *feature similarity* within each subgraph. Each fractal node summarizes the features of its corresponding subgraph by adaptively combining low-frequency (global) and high-frequency (local) components using a learnable parameter $\omega_c^{(\ell)}$. While mean pooling only retains global information through DC components, our approach preserves global patterns and local variations in the feature space.

Expressive power of *fractal nodes*. The expressive power of fractal nodes can be understood through the lens of existing theoretical results on subgraph-based approaches. The methods have been shown to increase expressive power beyond MPNNs. Encoding local subgraphs is stronger than 1-WL and 2-WL tests (Zhao et al. 2022, Theorem 4.3). In the context of the subgraph WL (SWL) test (Zhang et al. 2023), fractal nodes achieve expressive power comparable to SWL with additional single-point aggregation and potentially approach SWL with additional global aggregation (Zhang et al. 2023, Theorem 4.4), as the fractal nodes implicitly perform a form of global aggregation within each subgraph. We will empirically verify the expressive power in Sec. 5.2.

4.2 Model Complexity

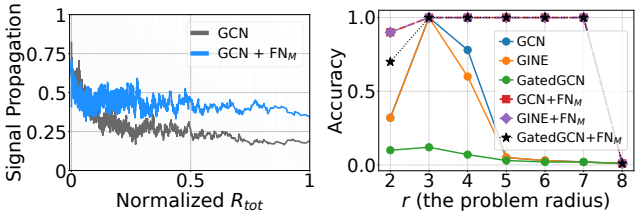
Our fractal nodes show improvements in computational efficiency compared to graph Transformers (Dwivedi and Bresson 2021; Rampášek et al. 2022). The time complexity of our FN is $\mathcal{O}(L(|\mathcal{V}| + |\mathcal{E}|))$, where L is the number of layers, $|\mathcal{V}|$ is the number of nodes, and $|\mathcal{E}|$ is the number of edges. The FN_M introduces an additional mixing step through the MLP-Mixer, leading to a time complexity of $\mathcal{O}(L(|\mathcal{V}|+|\mathcal{E}|)+Cd^2)$. C is the number of subgraphs and d is the hidden dimension. Given that C is smaller than $|\mathcal{V}|$, this term does not dominate the overall complexity. In contrast, graph Transformers incur a time complexity of $\mathcal{O}(L(|\mathcal{V}|^2))$, due to the quadratic cost of computing self-attention over all node pairs.

5 Experiments

To evaluate the effectiveness of our fractal nodes, we aim to answer the following key questions: **(Q1)** Can fractal nodes mitigate over-squashing in MPNNs? **(Q2)** Do fractal nodes improve the expressiveness of MPNNs? **(Q3)** How do fractal nodes compare to MPNNs and other graph Transformers regarding performance on benchmark datasets? **(Q4)** Does our method lead to a faster run time than graph Transformers? In this experiment, we aim to verify if fractal nodes provide meaningful benefits. Afterwards, we perform a series of additional studies, ablation studies, and sensitivity analyses.

5.1 Fractal Nodes Alleviate Over-squashing (Q1)

The signal propagation of MPNNs is inversely proportional to the total effective resistance (Di Giovanni et al. 2023). As shown in Fig. 3(a), while GCN fails to maintain signal flow under high total effective resistance, indicating severe bottlenecks, GCN+FN_M shows resilience by maintaining higher signal propagation even under the highest total effective resistance. This validates our theoretical predictions (Theorems 4.1 and 4.2) that fractal nodes reduce effective resistance by serving as single-hop shortcuts and enabling long-range interactions through the MLP-Mixer. We further evaluate TREENEIGHBOURSMATCH (Alon and Yahav 2021), where the task requires propagating information from leaf nodes to a target node in binary trees of depth r. As shown in Fig. 3(b), standard MPNNs fail for $r > 4$, while our methods generalize up to $r = 7$, empirically confirming that fractal nodes mitigate over-squashing.



(a) (b)

Figure 3: (a) The amount of signal propagated across the graph in PEPTIDES-FUNC. (b) Test accuracy across problem radius (tree depth) in the TREENEIGHBOURMATCH problem.

Method	CSL (Acc ↑)	SR25 (Acc ↑)	EXP(Acc ↑)
GCN	10.00	6.67	52.17
GCN + FN_M	39.67	100.0	86.40
GINE	10.00	6.67	51.35
GINE + FN_M	47.33	100.0	95.58
GatedGCN	10.00	6.67	51.25
GatedGCN + FN_M	49.67	100.0	96.50

Table 1: Results on 3 synthetic datasets

5.2 Expressive Power of Fractal Nodes (Q2)

We evaluate the expressive power of fractal nodes on 3 simulated datasets: CSL (Murphy et al. 2019), EXP (Abboud et al. 2021), and SR25 (Balcilar et al. 2021). Each dataset contains graphs indistinguishable by the 1 to 3-WL test, and details are provided in Appendix (Choi et al. 2025). Table 1 shows that our model achieves superior accuracy on all 3 datasets while MPNNs fail (see detailed result in Appendix (Choi et al. 2025)). Our results are empirical, but align with our discussion in Sec. 4.1.

5.3 Experiments on Graph Benchmarks (Q3)

Experimental setting and baselines. We evaluate our method on two tasks: graph-level prediction and large-scale node classification. For graph-level tasks, we use 6 benchmark datasets: 2 peptide datasets from LRGB (Dwivedi et al. 2022), 2 graph-level super-pixel image datasets from Benchmarking GNNs (Dwivedi et al. 2023), and 2 molecular datasets from OGB (Hu et al. 2020). For large-scale node classification, we use 2 large datasets from OGB (Hu et al. 2021): ogbn-arxiv and ogbn-products. We compare our fractal nodes to MPNNs, graph Transformer-based models, and other state-of-the-art models. Detailed experimental settings for graph-level tasks and large-scale node classification are provided in Appendix (Choi et al. 2025), respectively.

Results on graph-level tasks. Our proposed fractal nodes (FN and FN_M) consistently enhance the performance of baseline MPNNs on all benchmark datasets, often surpassing graph Transformer models. In Table 2, on PEPTIDES-FUNC, GINE+FN_M achieves an average precision (AP) of 0.7018, outperforming both Exphormer and GraphGPS. Our comparable performance with Graph-ViT and Exphormer on MNIST

Method	PEPTIDES-FUNC	PEPTIDES-STRUCT	MNIST	CIFAR10	MOLHIV	MOLTOX21
	AP ↑	MAE ↓	Accuracy ↑	Accuracy ↑	ROCAUC ↑	ROCAUC ↑
GCN (Kipf and Welling 2017)	0.6328±0.0023	0.2758±0.0012	0.9269±0.0023	0.5423±0.0056	0.7529±0.0098	0.7525±0.0031
GINE (Xu et al. 2019)	0.6405±0.0086	0.2780±0.0021	0.9705±0.0023	0.6131±0.0035	0.7885±0.0034	0.7730±0.0064
GatedGCN (Bresson and Laurent 2017)	0.6300±0.0029	0.2778±0.0017	0.9776±0.0017	0.6628±0.0017	0.7874±0.0119	0.7641±0.0057
GT (Dwivedi and Bresson 2021)	-	-	0.9083±0.0016	0.5975±0.0029	0.7350±0.0040	0.7500±0.0060
GraphiT (Mialon et al. 2021)	-	-	-	-	0.7460±0.0100	0.7180±0.0130
Graphormer (Ying et al. 2021)	-	-	-	-	0.7930±0.0040	0.7730±0.0800
Transformer + LapPE (Kreuzer et al. 2021)	0.6326±0.0126	0.2529±0.0016	0.9083±0.0016	0.5975±0.0029	-	0.7323±0.0057
SAN + LapPE (Kreuzer et al. 2021)	0.6384±0.0121	0.2683±0.0043	-	-	0.7775±0.0061	0.7130±0.0080
EGT (Hussain, Zaki, and Subramanian 2022)	-	-	0.9817±0.0009	0.6870±0.0041	-	-
GraphGPS (Rampášek et al. 2022)	0.6534±0.0091	0.2509±0.0014	0.9805±0.0013	0.7230±0.0036	0.7880±0.0101	0.7570±0.0040
GRIT (Ma et al. 2023)	**0.6988±0.0082**	0.2460±0.0012	0.9811±0.0011	**0.7647±0.0089**	-	-
Graph-ViT/MLP-Mixer (He et al. 2023)	0.6970±0.0080	**0.2449±0.0016**	**0.9846±0.0009**	0.7158±0.0009	0.7997±0.0102	**0.7910±0.0040**
Exphormer (Shirzad et al. 2023)	0.6527±0.0043	0.2481±0.0007	**0.9841±0.0035**	**0.7469±0.0013**	-	-
GECO (Sancak et al. 2024)	**0.6975±0.0025**	0.2464±0.0009	-	-	0.7980±0.0200	-
CRaWl (Tönshoff et al. 2023)	0.6963±0.0079	0.2506±0.0022	0.9794±0.0050	0.6901±0.0026	0.7707±0.1490	-
PNA (Corso et al. 2020)	-	-	0.9794±0.0012	0.7035±0.0063	0.7905±0.0132	-
GNN-AK+ (Zhao et al. 2022)	0.6480±0.0075	0.2736±0.0012	-	0.7219±0.0013	0.7961±0.0119	-
SUN (Frasca et al. 2022)	0.6730±0.0115	0.2498±0.0008	-	-	0.8003±0.0055	-
CIN (Bodnar et al. 2021)	-	-	-	-	**0.8094±0.0057**	-
GCN + FN	0.6802±0.0043	0.2530±0.0004	0.9393±0.0084	0.6006±0.0070	0.7564±0.0059	0.7670±0.0073
GINE + FN	0.6815±0.0059	0.2515±0.0020	0.9790±0.0012	0.6584±0.0069	0.7890±0.0104	0.7751±0.0029
GatedGCN + FN	0.6778±0.0056	0.2536±0.0019	0.9826±0.0012	0.7125±0.0035	0.7967±0.0098	0.7759±0.0054
GCN + FN$_M$	0.6787±0.0048	0.2464±0.0014	0.9455±0.0004	0.6413±0.0068	0.7866±0.0034	0.7882±0.0041
GINE + FN$_M$	**0.7018±0.0074**	**0.2446±0.0018**	0.9786±0.0004	0.6672±0.0068	**0.8127±0.0076**	**0.7926±0.0021**
GatedGCN + FN$_M$	0.6950±0.0047	**0.2453±0.0014**	**0.9848±0.0005**	**0.7526±0.0033**	**0.8097±0.0047**	**0.7922±0.0054**

Table 2: Test performance on 6 benchmark datasets. The top three models are colored by **first**, **second**, **third**.

Method	ogbn-arxiv	ogbn-product
# nodes	169,343	2,449,029
# edges	1,166,243	61,859,140
GraphGPS (Rampášek et al. 2022)	70.97±0.41	OOM
NAGphormer (Chen et al. 2023)	70.13±0.55	73.55±0.21
Exphormer (Shirzad et al. 2023)	72.44±0.28	OOM
NodeFormer (Wu et al. 2022)	69.86±0.25	72.93±0.13
DiffFormer (Wu et al. 2023a)	72.41±0.40	74.16±0.31
PolyNormer (Deng, Yue, and Zhang 2024)	71.82±0.23	**82.97±0.28**
SGFormer (Wu et al. 2023b)	72.63±0.13	74.16±0.31
HC-GNN (Zhong, Li, and Pang 2023)	72.79±0.25	-
ANS-GT (Cai, Wang, and Wang 2021)	72.34±0.50	80.64±0.29
HSGT (Zhu et al. 2023)	72.58±0.31	81.15±0.13
GCN	71.74±0.29	75.64±0.21
GCN + FN	**73.03±0.37**	81.29±0.21
GCN + FN$_M$	**72.93±0.35**	81.33±0.33
GraphSAGE	71.49±0.27	78.29±0.16
GraphSAGE + FN	**72.70±0.11**	**83.07±0.35**
GraphSAGE + FN$_M$	72.54±0.30	**83.11±0.07**

Table 3: Results on large-scale graphs (%). OOM means out of memory.

shows that fractal nodes can effectively capture local and global information without a self-attention layer.

Results on large-scale graphs. The effectiveness of our method is particularly evident in large-scale graph experiments in Table 3. On ogbn-arxiv, GCN+FN improves accuracy from 71.74% to 73.03%, while on ogbn-product, GraphSAGE+FN$_M$ demonstrates a substantial improvement. Exphormer and GraphGPS fail to scale to ogbn-products due to their complexity in attention computation. In contrast, our method maintains computational efficiency while achieving superior performance, which we discuss further in Sec. 5.4. This highlights the effectiveness of fractal nodes in capturing local and graph information and their practical applicability to large-scale graphs.

5.4 Runtime Comparison (Q4)

As discussed in Sec. 4.2, our fractal nodes benefit from capturing long-range dependencies without increasing computational complexity. As shown in Table 5, fractal nodes introduce trivial computational overhead – GCN+FN maintains identical training time and memory usage compared to the GCN. Our method uses standard MPNN operations without introducing additional, complex computations. In contrast, graph Transformers (e.g., GraphGPS and Exphormer) require more computational resources due to their attention mechanisms. Given these results shown in Secs. 5.3 and 5.4, we believe our method achieves a balance between accuracy and computational efficiency.

In Appendix (Choi et al. 2025), we also conduct experi-

Method	PEPTIDES-FUNC	PEPTIDES-STRUCT	MOLHIV	MOLTOX21
	AP ↑	MAE ↓	ROCAUC ↑	ROCAUC ↑
GCN + virtual node (Hu et al. 2020)	0.6455±0.0020	0.2745±0.0013	0.7599±0.0119	0.7551±0.0100
GIN + virtual node (Hu et al. 2020)	-	-	0.7707±0.0149	0.7621±0.0062
GCN + VN (Rosenbluth et al. 2024)	0.6732±0.0066	0.2505±0.0022	-	-
GatedGCN + VN (Rosenbluth et al. 2024)	**0.6823**±0.0069	0.2475±0.0018	-	-
GatedGCN + VN_G (Southern et al. 2025)	0.6822±0.0052	0.2458±0.0006	**0.7910**±0.0086	0.7655±0.0060
GCN + FN_M	0.6787±0.0048	**0.2464**±0.0014	0.7866±0.0034	**0.7882**±0.0041
GINE + FN_M	**0.7018**±0.0074	**0.2446**±0.0018	**0.8127**±0.0076	**0.7926**±0.0021
GatedGCN + FN_M	**0.6950**±0.0047	**0.2453**±0.0014	**0.8097**±0.0047	**0.7922**±0.0054

Table 4: Comparison to virtual node methods.

Method	Runtime (s)	Mem. (GB)
GCN	1.27	16.49
GraphGPS	1.32	38.91
Exphormer	0.74	34.04
NodeFormer	1.20	16.30
DiffFormer	0.77	24.51
PolyNormer	0.31	16.09
GCN + FN	1.27	16.49
GCN + FN_M	1.27	16.49
GraphSAGE + FN	0.57	7.74
GraphSAGE + FN_M	0.58	7.76

Table 5: Training time per epoch and memory usage on ogbn-arxiv.

Method	PEP.-FUNC	PEP.-STRUCT
	AP ↑	MAE ↓
GCN	0.5930±0.0023	0.3496±0.0013
+ FoSR (Karhadkar et al. 2023)	0.5947±0.0035	0.3473±0.0007
+ GTR (Black et al. 2023)	0.5075±0.0029	0.3618±0.0010
+ SDRF (Topping et al. 2022)	0.5947±0.0126	0.3478±0.0013
+ BORF (Nguyen et al. 2023)	0.5994±0.0037	0.3514±0.0009
+ PANDA (Choi et al. 2024a)	**0.6028**±0.0031	**0.3272**±0.0001
+ LASER (Barbero et al. 2023)	**0.6440**±0.0010	**0.3043**±0.0019
+ FN	**0.6445**±0.0057	**0.2535**±0.0012

Table 6: Results of GCN with rewiring methods.

ments on synthetic Erdős-Rényi graphs with nodes ranging from 1,000 to 100,000, demonstrating linear space complexity in GPU memory usage and showing that METIS with $\mathcal{O}(|\mathcal{E}|)$ complexity enables efficient fractal node creation.

5.5 Ablation, Sensitivity, and Additional Studies

We report ablation studies for $\omega_c^{(\ell)}$ and HPF in Appendix (Choi et al. 2025). We report results when $\omega_c^{(\ell)}$ is zero, that is, without HPF, and when we use either a scalar parameter or a learnable vector parameter. We provide sensitivity studies on the number of fractal nodes C, additional analyses on message passing variants and the distribution of subgraph size ratios, and evaluations of other partitioning algorithms (random, Louvain (Blondel et al. 2008), Girvan-Newman (Girvan and Newman 2002)) in Appendix (Choi et al. 2025).

Comparison with augmented MPNNs. We compare fractal nodes against two categories of augmented MPNNs: graph rewiring and virtual node approaches. Table 6 compares our method with 6 methods on 2 PEPTIDES datasets. We replicate the settings of Dwivedi et al. (2022) and use the results from Barbero et al. (2023). All methods use the same 500k parameter budget, without positional encodings, for a fair comparison. Our FN achieves competitive performance, outperforming all baselines, including LASER. When using a single fractal node, our method conceptually reduces to a virtual node that aggregates global information. However, as shown in Table 4, FN_M outperforms all virtual node methods. Particularly, GINE+FN_M achieves the best results on all metrics. This shows that our frequency-based aggregation provides more effective representations than virtual nodes.

6 Concluding Remark

We introduced *fractal nodes* that preserve the computational efficiency of MPNNs while allowing long-range interactions that graph Transformers typically capture. Our core idea is that graph partitioning naturally induces *fractal structure*, and fractal nodes enforce *feature similarity* within each subgraph. This design alleviates over-squashing and improves expressive power through subgraph-level representations. Experiments show that fractal nodes consistently improve MPNN performance and achieve competitive results with graph Transformers.

Limitations and future directions. While fractal nodes are effective, they rely on graph partitioning, which may not capture optimal clustering for all graph types. While we show robustness across different partitioning methods in Appendix (Choi et al. 2025), learnable partitioning could improve performance. The fixed number of fractal nodes also limits adaptability to varying graph structures. Despite these limitations, our extensive experiments demonstrate that fractal nodes offer a practical and effective alternative to graph Transformers for most real-world applications.

Acknowledgements

Noseong Park is the corresponding author. This work was partly supported by the Institute for Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korean government (MSIT) (No. RS-2024-00457882, AI Research Hub Project; No. RS-2025-25442149, LG AI STAR Talent Development Program for Leading Large-Scale Generative AI Models in the Physical AI Domain), Samsung Electronics Co., Ltd. (No. G01240136, KAIST Semiconductor Research Fund (2nd)), and an ETRI grant funded by the Korean government (No. 24ZB1100, Core Technology Research for Self-Improving Integrated Artificial Intelligence System).

References

Abboud, R.; Ceylan, İ. İ.; Grohe, M.; and Lukasiewicz, T. 2021. The Surprising Power of Graph Neural Networks with Random Node Initialization. In *IJCAI*.

Alon, U.; and Yahav, E. 2021. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *ICLR*.

Balcilar, M.; Héroux, P.; Gauzere, B.; Vasseur, P.; Adam, S.; and Honeine, P. 2021. Breaking the limits of message passing graph neural networks. In *ICML*, 599–608. PMLR.

Barbero, F.; Velingker, A.; Saberi, A.; Bronstein, M.; and Di Giovanni, F. 2023. Locality-Aware Graph-Rewiring in GNNs. *arXiv preprint arXiv:2310.01668*.

Black, M.; Wan, Z.; Nayyeri, A.; and Wang, Y. 2023. Understanding Oversquashing in GNNs through the Lens of Effective Resistance. In *ICML*.

Blondel, V. D.; Guillaume, J.-L.; Lambiotte, R.; and Lefebvre, E. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10): P10008.

Bodnar, C.; Frasca, F.; Otter, N.; Wang, Y.; Lio, P.; Montufar, G. F.; and Bronstein, M. 2021. Weisfeiler and lehman go cellular: Cw networks. *NeurIPS*, 34: 2625–2640.

Bresson, X.; and Laurent, T. 2017. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*.

Cai, C.; Hy, T. S.; Yu, R.; and Wang, Y. 2023. On the connection between mpnn and graph transformer. In *ICML*.

Cai, C.; Wang, D.; and Wang, Y. 2021. Graph Coarsening with Neural Networks. In *ICLR*.

Chen, J.; Gao, K.; Li, G.; and He, K. 2023. NAGphormer: A Tokenized Graph Transformer for Node Classification in Large Graphs. In *ICLR*.

Chen, Y.; Li, R.; Zhao, Z.; and Zhang, H. 2020. On the capacity of fractal D2D social networks with hierarchical communications. *IEEE Transactions on Mobile Computing*.

Choi, J.; Hong, S.; Park, N.; and Cho, S.-B. 2023. GREAD: Graph neural reaction-diffusion networks. In *ICML*.

Choi, J.; Jeon, J.; and Park, N. 2021. LT-OCF: Learnable-Time ODE-based Collaborative Filtering. In *CIKM*.

Choi, J.; Park, S.; Park, S.; Cho, S.-B.; and Park, N. 2025. Are Graph Transformers Necessary? Efficient Long-Range Message Passing with Fractal Nodes in MPNNs. *arXiv preprint arXiv:2511.13010*.

Choi, J.; Park, S.; Wi, H.; Cho, S.-B.; and Park, N. 2024a. PANDA: Expanded Width-Aware Message Passing Beyond Rewiring. In *ICML*.

Choi, J.; Wi, H.; Kim, J.; Shin, Y.; Lee, K.; Trask, N.; and Park, N. 2024b. Graph Convolutions Enrich the Self-Attention in Transformers! In *NeurIPS*.

Corso, G.; Cavalleri, L.; Beaini, D.; Liò, P.; and Veličković, P. 2020. Principal neighbourhood aggregation for graph nets. *NeurIPS*, 33: 13260–13271.

Deng, C.; Yue, Z.; and Zhang, Z. 2024. Polynormer: Polynomial-Expressive Graph Transformer in Linear Time. In *ICLR*.

Di Giovanni, F.; Giusti, L.; Barbero, F.; Luise, G.; Lio, P.; and Bronstein, M. M. 2023. On Over-Squashing in Message Passing Neural Networks: The Impact of Width, Depth, and Topology. In *ICML*.

Dill, S.; Kumar, R.; McCurley, K. S.; Rajagopalan, S.; Sivakumar, D.; and Tomkins, A. 2002. Self-similarity in the web. *ACM Transactions on Internet Technology*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*.

Dwivedi, V. P.; and Bresson, X. 2021. A Generalization of Transformer Networks to Graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*.

Dwivedi, V. P.; Joshi, C. K.; Luu, A. T.; Laurent, T.; Bengio, Y.; and Bresson, X. 2023. Benchmarking graph neural networks. *Journal of Machine Learning Research*.

Dwivedi, V. P.; Rampášek, L.; Galkin, M.; Parviz, A.; Wolf, G.; Luu, A. T.; and Beaini, D. 2022. Long Range Graph Benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Frasca, F.; Bevilacqua, B.; Bronstein, M.; and Maron, H. 2022. Understanding and Extending Subgraph GNNs by Rethinking Their Symmetries. In *NeurIPS*, 31376–31390.

Freeman, L. C. 1977. A set of measures of centrality based on betweenness. *Sociometry*, 35–41.

Gasteiger, J.; Weißenberger, S.; and Günnemann, S. 2019. Diffusion improves graph learning. In *NeurIPS*.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*.

Girvan, M.; and Newman, M. E. 2002. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12): 7821–7826.

He, X.; Hooi, B.; Laurent, T.; Perold, A.; LeCun, Y.; and Bresson, X. 2023. A generalization of vit/mlp-mixer to graphs. In *ICML*.

Hu, W.; Fey, M.; Ren, H.; Nakata, M.; Dong, Y.; and Leskovec, J. 2021. OGB-LSC: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open graph benchmark: Datasets for machine learning on graphs. *NeurIPS*.

Hussain, M. S.; Zaki, M. J.; and Subramanian, D. 2022. Global self-attention as a replacement for graph convolution. In *KDD*, 655–665.

Jeong, Y. U.; Choi, J.; Park, N.; Ryu, J. Y.; and Kim, Y. R. 2025. Predicting drug-drug interactions: a deep learning approach with GCN-based collaborative filtering. *Artificial Intelligence in Medicine*, 103185.

Karhadkar, K.; Banerjee, P. K.; and Montufar, G. 2023. FoSR: First-order spectral rewiring for addressing oversquashing in GNNs. In *ICLR*.

Karypis, G.; and Kumar, V. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1): 359–392.

Kim, P.; and Kahng, B. 2010. Fractal Network in Protein Interaction Network Model. *Journal of the Korean Physical Society*, 56(3): 1020–1024.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Kitsak, M.; Havlin, S.; Paul, G.; Riccaboni, M.; Pammolli, F.; and Stanley, H. E. 2007. Betweenness centrality of fractal and nonfractal scale-free model networks and tests on real networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 75(5): 056115.

Kreuzer, D.; Beaini, D.; Hamilton, W.; Létourneau, V.; and Tossou, P. 2021. Rethinking Graph Transformers with Spectral Attention. In *NeurIPS*.

Ma, L.; Lin, C.; Lim, D.; Romero-Soriano, A.; Dokania, P. K.; Coates, M.; Torr, P.; and Lim, S.-N. 2023. Graph inductive biases in transformers without message passing. In *ICML*.

Mandelbrot, B. B. 1983. The fractal geometry of nature/Revised and enlarged edition. *New York*.

Mialon, G.; Chen, D.; Selosse, M.; and Mairal, J. 2021. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*.

Murphy, R.; Srinivasan, B.; Rao, V.; and Ribeiro, B. 2019. Relational pooling for graph representations. In *ICML*.

Nguyen, K.; Hieu, N. M.; Nguyen, V. D.; Ho, N.; Osher, S.; and Nguyen, T. M. 2023. Revisiting Over-smoothing and Over-squashing Using Ollivier-Ricci Curvature. In *ICML*.

Nt, H.; and Maehara, T. 2019. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*.

Rampášek, L.; Galkin, M.; Dwivedi, V. P.; Luu, A. T.; Wolf, G.; and Beaini, D. 2022. Recipe for a general, powerful, scalable graph transformer. *NeurIPS*, 35: 14501–14515.

Rosenbluth, E.; Tönshoff, J.; Ritzert, M.; Kisin, B.; and Grohe, M. 2024. Distinguished In Uniform: Self-Attention Vs. Virtual Nodes. In *ICLR*.

Sancak, K.; Hua, Z.; Fang, J.; Xie, Y.; Malevich, A.; Long, B.; Balin, M. F.; and Çatalyürek, Ü. V. 2024. A Scalable and Effective Alternative to Graph Transformers. *arXiv preprint arXiv:2406.12059*.

Shirzad, H.; Velingker, A.; Venkatachalam, B.; Sutherland, D. J.; and Sinop, A. K. 2023. Exphormer: Sparse transformers for graphs. In *ICML*.

Song, C.; Havlin, S.; and Makse, H. A. 2005. Self-similarity of complex networks. *Nature*, 433(7024): 392–395.

Southern, J.; Giovanni, F. D.; Bronstein, M. M.; and Lutzeyer, J. F. 2025. Understanding Virtual Nodes: Oversquashing and Node Heterogeneity. In *ICLR*.

Tolstikhin, I. O.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Steiner, A.; Keysers, D.; Uszkoreit, J.; et al. 2021. Mlp-mixer: An all-mlp architecture for vision. *NeurIPS*, 34: 24261–24272.

Tönshoff, J.; Ritzert, M.; Wolf, H.; and Grohe, M. 2023. Walking Out of the Weisfeiler Leman Hierarchy: Graph Learning Beyond Message Passing. *TMLR*.

Topping, J.; Giovanni, F. D.; Chamberlain, B. P.; Dong, X.; and Bronstein, M. M. 2022. Understanding over-squashing and bottlenecks on graphs via curvature. In *ICLR*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NeurIPS*, volume 30.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*.

Wu, Q.; Yang, C.; Zhao, W.; He, Y.; Wipf, D.; and Yan, J. 2023a. DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion. In *ICLR*.

Wu, Q.; Zhao, W.; Li, Z.; Wipf, D. P.; and Yan, J. 2022. Nodeformer: A scalable graph structure learning transformer for node classification. *NeurIPS*, 35: 27387–27401.

Wu, Q.; Zhao, W.; Yang, C.; Zhang, H.; Nie, F.; Jiang, H.; Bian, Y.; and Yan, J. 2023b. Simplifying and empowering transformers for large-graph representations. *NeurIPS*, 36.

Wu, Z.; Jain, P.; Wright, M.; Mirhoseini, A.; Gonzalez, J. E.; and Stoica, I. 2021. Representing long-range context for graph neural networks with global attention. *NeurIPS*.

Xing, Y.; Wang, X.; Li, Y.; Huang, H.; and Shi, C. 2024. Less is More: on the Over-Globalizing Problem in Graph Transformers. In *ICML*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *ICLR*.

Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; and Liu, T.-Y. 2021. Do transformers really perform badly for graph representation? In *NeurIPS*, volume 34, 28877–28888.

Yu, Y.-Y.; Choi, J.; Park, J.; Lee, K.; and Park, N. 2025. PI-ORF: Physics-Informed Ollivier-Ricci Flow for Long–Range Interactions in Mesh Graph Neural Networks. In *ICLR*.

Zhang, B.; Feng, G.; Du, Y.; He, D.; and Wang, L. 2023. A complete expressiveness hierarchy for subgraph gnns via subgraph weisfeiler-lehman tests. In *ICML*.

Zhao, L.; Jin, W.; Akoglu, L.; and Shah, N. 2022. From Stars to Subgraphs: Uplifting Any GNN with Local Structure Awareness. In *ICLR*.

Zhong, Z.; Li, C.-T.; and Pang, J. 2023. Hierarchical message-passing graph neural networks. *Data Mining and Knowledge Discovery*, 37(1): 381–408.

Zhu, W.; Wen, T.; Song, G.; Ma, X.; and Wang, L. 2023. Hierarchical transformer for scalable graph learning. In *IJCAI*.