SCALABLE MESSAGE PASSING NEURAL NETWORKS: NO NEED FOR ATTENTION IN LARGE GRAPH REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose Scalable Message Passing Neural Networks (SMPNNs) and demonstrate that, by integrating standard convolutional message passing into a Pre-Layer Normalization Transformer-style block instead of attention, we can produce high-performing deep message-passing-based Graph Neural Networks (GNNs). This modification yields state-of-the-art results in large graph transductive learning, outperforming the best Graph Transformers in the literature without requiring the otherwise computationally and memory-expensive attention. Our architecture not only scales to large graphs but also makes it possible to construct deep message-passing networks, unlike simple GNNs, which have traditionally been constrained to shallow architectures due to oversmoothing. Moreover, we provide a new theoretical analysis of oversmoothing based on universal approximation which we use to motivate SMPNNs.

1 INTRODUCTION

Traditionally, Graph Neural Networks (GNNs) (Scarselli et al., 2009) have primarily been applied to model functions over graphs with a relatively modest number of nodes. However, recently there has been a growing interest in exploring the application of GNNs to large-scale graph benchmarks, including datasets with up to a hundred million nodes (Hu et al., 2020). This exploration could potentially lead to better models for industrial applications such as large-scale network analysis in social media, where there are typically millions of users, or in biology, where proteins and other macromolecules are composed of a large number of atoms. This presents a significant challenge in designing GNNs that are scalable while retaining their effectiveness.

To this end, we take inspiration from the literature on Large Language Models (LLMs) and propose a simple modification to how GNN architectures are typically designed. Our framework, Scalable Message Passing Neural Networks (SMPNNs), enables the construction of deep and scalable architectures that outperform the current state-of-the-art models for large graph benchmarks in transductive node classification. More specifically, we find that following the typical construction of the Pre-Layer Normalization (Pre-LN) Transformer formulation (Xiong et al., 2020) and replacing attention with standard message-passing convolution are enough to outperform the best Graph Transformers in the literature. Moreover, since our formulation does not necessarily require attention, our architecture scales better than Graph Transformers. Attention can also be easily incorporated into our framework if needed; however, in general, we find that adding attention, at least for large-scale graph transductive learning, only leads to marginal improvements in performance at the cost of being more computationally demanding.

Our empirical observations, which demonstrate that SMPNNs can use many layers unlike traditional GNNs, are supported by recent theoretical studies on oversmoothing and oversharpening in graph convolutions (Giovanni et al., 2023) and Transformers (Dovonon et al., 2024). These studies suggest the crucial role of residual connections in mitigating oversmoothing and low-frequency dominance in representations. It is worth noting, however, that the aforementioned works primarily approached this issue from a theoretical standpoint and did not scale to large graph datasets. Expanding upon previous theoretical studies on oversmoothing, we provide a universal approximation perspective. Specifically, we demonstrate that residual connections, such as those found in the Transformer block architecture and other newer blocks utilized in the LLM literature, like the Mamba block (Gu & Dao, 2023), are essential for preserving the universal approximation properties of downstream classifiers.

Contributions. We propose *Scalable Message Passing Neural Networks* (SMPNNs), a framework designed to scale traditional message-passing GNNs. Our main contributions are the following:

(i) The SMPNN architecture can *scale to large graphs* thanks to its $\mathcal{O}(E)$ graph convolution computational complexity, outperforming state-of-the-art Graph Transformers for transductive learning without using global attention mechanisms. It also enables *deep message-passing* GNNs without suffering from oversmoothing, a problem that has traditionally limited these networks to shallow configurations.

(ii) We theoretically analyze the advantages of our architecture compared to traditional convolutions over graphs from a universal approximation perspective. Importantly, unlike previous works, we do not rely on the asymptotic convergence of the system to explain its behaviour.

(iii) We perform extensive experiments in large-graph transductive learning as well as on smaller datasets and demonstrate that our model consistently outperforms recently proposed Graph Transformers and other traditional scalable architectures. We also conduct ablations and additional experiments to test the importance of the different components of our architecture and support our theoretical claims.

2 BACKGROUND

Graph Representation Learning and Notation. We denote a graph with $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents a set of nodes (or vertices), and $\mathcal{E} \subseteq (\mathcal{V} \times \mathcal{V})$ is a set of 2-tuples signifying edges (or links) within the graph. For any pair of nodes v_i and v_j within $\mathcal{G}$, their connection is encoded as $(v_i, v_j) \in \mathcal{E}$ if the edge originates from v_i and terminates at v_j. The (one-hop) neighborhood of node v_i, constitutes the set of nodes sharing an edge with v_i, denoted by $\mathcal{N}(v_i) = \{v_j | (v_i, v_j) \in \mathcal{E}\}$. To encode the structural connectivity amongst nodes in a graph of $N = |\mathcal{V}|$ nodes and $E = |\mathcal{E}|$ edges, an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ is employed. This adjacency matrix may adopt a weighted or unweighted representation. In the case of a weighted adjacency matrix, the entries $A_{ij} \in \mathbb{R}$ symbolize the strength of the connection, with $A_{ij} = 0$ denoting absence of a connection if $(v_i, v_j) \notin \mathcal{E}$. Conversely, in an unweighted adjacency matrix, $A_{ij} = 1$ signifies the presence of an edge, and $A_{ij} = 0$ otherwise. The degree matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$ is defined as the matrix where each entry on the diagonal is the row-sum of the adjacency matrix: $D_{ii} = \sum_j A_{ij}$. Based on this, we can define the graph Laplacian as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. The normalized graph Laplacian, denoted as $\mathbf{L}_{\text{norm}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, is a variation of the graph Laplacian that takes into account the degrees of the nodes. Additionally, within our context, we are interested in graphs with node features. Each node $v_i \in \mathcal{V}$ is accompanied by a D-dimensional feature vector $\mathbf{x}_i \in \mathbb{R}^{1 \times D}$. The feature vectors for all nodes within the graph can be aggregated into a single matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ by stacking them along the row dimension.

Message Passing Neural Networks (MPNNs) are a class of Graph Neural Networks (GNNs). MPNNs operate by means of message passing, wherein nodes exchange vector-based messages to refine their representations, leveraging the graph connectivity structure as a geometric prior. Following Bronstein et al. (2021), a message passing GNN layer l (excluding edge and graph-level features for simplicity) over a graph $\mathcal{G}$ is defined as:

$$\mathbf{x}_i^{(l+1)} = \phi\left(\mathbf{x}_i^{(l)}, \bigoplus_{j \in \mathcal{N}(v_i)} \psi(\mathbf{x}_i^{(l)}, \mathbf{x}_j^{(l)})\right), \tag{1}$$

where ψ denotes a message passing function, $\bigoplus$ is some permutation-invariant aggregation operator, and ϕ is a readout or update function. Both ψ and ϕ are learnable and typically implemented as MLPs. It is important to note that the update equation is *local*, and that at each layer each node only communicates with its one-hop neighborhood given by the adjacency matrix of the graph. More distant nodes are accessed by stacking multiple message-passing layers.

Graph Convolutional Networks (GCNs) implement a special type of message passing of the following form:

$$\mathbf{X}^{(l+1)} = \zeta\left(\tilde{\mathbf{A}}\mathbf{X}^{(l)}\mathbf{W}^{(l)}\right), \tag{2}$$

where ζ is a non-linear activation function, $\mathbf{W}^{(l)}$ is a learnable weight matrix, and $\tilde{\mathbf{A}}$ is computed as a function of the degree matrix and the adjacency matrix. The features are aggregated row-wise:

$$\left(\tilde{\mathbf{A}}\mathbf{X}^{(l)}\right)_i = c_{ii}\mathbf{x}_i^{(l)} + \sum_{j \in \mathcal{N}(v_i)} c_{ij}\mathbf{x}_j^{(l)} \,;\, c_{ij} = \frac{1}{\sqrt{\deg(v_i)\deg(v_j)}}. \tag{3}$$

Transformers. The Transformer architecture (Vaswani et al., 2017b) is characterized by its all-to-all pairwise communication between nodes via the scaled dot-product attention mechanism. Transformers can be seen as an MPNN on a fully-connected graph (Sáez de Ocáriz Borde, 2024; Bronstein et al., 2021), where all nodes are within the one-hop neighborhood of each other in every layer. This avoids issues in capturing long-range dependencies inherent to message passing, since they are all connected. However, Transformers discard the locality inductive bias of the input graph. Given input query, key, and value matrices, $\mathbf{Q} = \mathbf{X}\mathbf{W}_Q$, $\mathbf{K} = \mathbf{X}\mathbf{W}_K$, $\mathbf{V} = \mathbf{X}\mathbf{W}_V \in \mathbb{R}^{N \times D}$ (where for simplicity we assume the feature dimensionality of these matrices is kept the same as the original node features, $\mathbf{X} \in \mathbb{R}^{N \times D}$), the self-attention operation in the encoder blocks computes:

$$\mathbf{X}' = \text{softmax}(\mathbf{Q}\mathbf{K}^T/\sqrt{D})\mathbf{V}. \tag{4}$$

Attention is challenging to scale to large graphs due to its computational complexity of $\mathcal{O}(N^2)$ in the number of nodes. Large-scale Graph Transformers typically replace this operation with linear attention to avoid GPU memory overflow.

Large Graph Transformers. The emergence of large-scale graph-structured datasets such as social networks has led to increased recent interest in scaling GNNs to very large graphs with up to hundreds of millions of nodes. Large-scale graph learning settings are often transductive learning, where one is given a fixed graph with partially labeled nodes and attempts to infer the properties of the missing nodes. While Graph Transformers currently produce state-of-the-art results, their main challenge is the computational complexity of attention for a large number of nodes, which is equivalent to very long context windows in LLMs. Techniques aiming at solving this problem include architectures such as Nodeformer (Wu et al., 2022), which utilizes a kernelized Gumbel-Softmax operator, Difformer (Wu et al., 2023a), which builds on recent advances in graph diffusion, and SGFormer (Wu et al., 2023b), which utilizes a single linear "attention" operation without the softmax activation to avoid computational overhead when scaling to graphs with up to a hundred million nodes. Other architectures such as Exphormer (Shirzad et al., 2023) use sparse attention mechanisms based on expander graphs to aid scaling. Additionally, Exphormer also uses linear global attention with respect to a virtual node in parallel to the expander graph. Some of the aforementioned architectures rely on the GPS Graph Transformer framework (Rampášek et al., 2022), which combines attention with standard message-passing GNN layers and computes both in parallel in the form of a hybrid architecture. Other scalable architectures that are not based on Transformers, such as SGC (Wu et al., 2019) and SIGN (Frasca et al., 2020), have been developed for large-scale graphs, but their performance is generally worse than that of Graph Transformers, as shown in Section 5.

3 Scalable Message Passing Neural Networks

3.1 Motivation

In the GNN literature, it has been shown that graph convolutions can exhibit asymptotic behaviors other than over-smoothing in the limit of many layers when equipped with residual connections (Giovanni et al., 2023). This theoretical observation aligns with well-known best practices in the LLM community (Vaswani et al., 2017b; Touvron et al., 2023; Brown et al., 2020), which is highly experienced with both large-scale datasets and models. In this work, we aim to bridge the gap between the GNN and LLM literature and benefit from the cross-pollination of ideas.

Packaging Attention and Message-Passing. Attention has been much emphasized in the literature and, although it is certainly paramount, the importance of the other components around it should not be underestimated. Interestingly, in retrospect, the ubiquitous attention mechanism used in LLMs had already been proposed (Bahdanau et al., 2016) before the Transformer architecture (Vaswani et al., 2017a); however, it was not until it was "packaged" into the Transformer block that we know today, that it really led to the recent breakthroughs in language modeling. The scaled-dot-product attention mechanism (and its variants) in Transformers can be seen as message-passing attentional GNNs over a complete graph (Sáez de Ocáriz Borde, 2024; Bronstein et al., 2021). It is well-known among practitioners in the LLM community that this operation struggles to learn effectively without residual connections. Yet, traditional GNN architectures simply stack message-passing layers one after the other (Kipf & Welling, 2017a; Schlichtkrull et al., 2017; Veličković et al., 2018a), which seems to contradict the modus operandi of large-scale model engineering.

The main motivation of our paper is to demonstrate that the standard architectural packaging approaches used in NLP and LLMs are also applicable to GNNs, and that the standard attention mechanism used in Transformers can be substituted with message-passing layers.

3.2 Architecture Design and the Scalable Message Passing Block

Drawing from theoretical observations in the GNN literature and best practices in language modeling, we propose the following architecture, which is similar to the Transformer and comprises two parts: initial nodewise message-passing, analogous to tokenwise communication, and a pointwise feedforward layer to transform the feature vector of each node in isolation. Unlike the Transformer, however, our architecture has linear rather than quadratic scaling.

The Transformer Block. The original Transformer architecture has proven surprisingly robust, and the only main modification it has undergone since its inception is the change in the location of the normalization, which is now applied before attention. Indeed, the advantages of applying layer normalization (Ba et al., 2016) before, as in the Pre-LN Transformer, have already been studied both theoretically (Xiong et al., 2020) and empirically in the literature (Baevski & Auli, 2019; Child et al., 2019; Wang et al., 2019), and we will also follow this in our implementation. Before the Transformer, residual connections were already a common component in most deep architectures for vision and language tasks since being popularized by the ResNet model (He et al., 2015). In the LLM literature, even when alternatives to attention have been proposed, such as Mamba (Gu & Dao, 2023), operations that perform token-wise communication (equivalent to message-passing between nodes) are always accompanied by *residual connections*, as well as *linear projections*, *normalization*, and sometimes *gatings*.

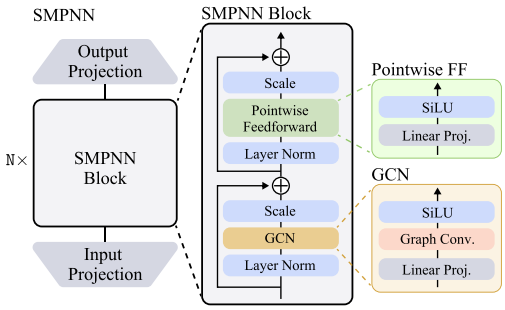

Figure 1: **The Scalable Message Passing Neural Network (SMPNN) architecture.** *Left:* The full model is comprised of N transformer-style blocks stacked one after the other. The model also uses input and output feedforward layers to project node features to the hidden and output dimensions. *Middle:* Architecture of a single SMPNN block as described in Section 3.2. *Right:* Zoom into the GCN block and the Pointwise FeedForward network with SiLU activation functions.

Message-Passing. We integrate a standard GCN layer into the Pre-LN Transformer style block. In particular, a single block applies the following sequence of transformations to the input matrix of node features $\mathbf{X}^{(l)} \in \mathbb{R}^{N \times D}$, where N is the number of nodes and D is the dimensionality of the feature vectors:

$$\mathbf{H}_1^{(l)} = \text{LayerNorm}(\mathbf{X}^{(l)}). \tag{5}$$

Next a GCN layer (plus additional components) is used for nodewise local communication instead of the standard global self-attention:

$$\mathbf{H}_2^{(l)} = \alpha_1^{(l)} \,\text{SiLU}\left(\tilde{\mathbf{A}}\mathbf{H}_1^{(l)}\mathbf{W}_1^{(l)}\right) + \mathbf{X}^{(l)}, \tag{6}$$

where $\tilde{\mathbf{A}}$, the degree-normalized adjacency matrix is $\tilde{\mathbf{A}}_{i,j} = 1/\sqrt{\deg(v_i)\deg(v_j)}$ if v_i and v_j are neighbours and 0 otherwise. $\text{SiLU}(x) = \frac{x}{1+e^{-x}}$ and it is applied elementwise. We also introduce a scaling factor $\alpha_1^{(l)}$ which is initialized at 10^{-6} for applying identity-style block initialization (Peebles & Xie, 2023). Importantly, the layer defined in equation 6 contains a residual connection, the importance of which will be theoretically justified in Section 4.

Pointwise-feedforward. The second part of the block is a pointwise transformation of the feature vectors preceded by another learnable normalization:

$$\mathbf{H}_3^{(l)} = \text{LayerNorm}(\mathbf{H}_2^{(l)}), \tag{7}$$

$$\mathbf{X}^{(l+1)} = \mathbf{H}_4^{(l)} = \alpha_2^{(l)} \,\text{SiLU}(\mathbf{H}_3^{(l)}\mathbf{W}_2^{(l)}) + \mathbf{H}_2^{(l)}, \tag{8}$$

where similar to before, we introduce the learnable scaling $\alpha_2^{(l)}$, also initialized at 10^{-6}.

Our *Scalable Message Passing Neural Network block* is thus of the form $\mathbf{X} \mapsto \mathbf{H}_4$. In Figure 1, we display the complete SMPNN architecture at different levels of granularity, which consists of stacking multiple instances of the aforementioned block. In Appendix A, we discuss how to augment SMPNNs with attention, although we find that this only leads to minor performance improvements, see Table 2 in Section 5.

3.3 Computational Complexity and Comparison with Graph Transformers

In Table 1, we compare the computational complexity of SMPNNs to that of various Graph Transformers in the literature. Our model's graph convolution layer inherits the computational cost of GCNs, $\mathcal{O}(E)$, assuming a sparse representation of the adjacency matrix (Kipf & Welling, 2017b). Graph Transformers such as SGFormer use both graph convolutions and linear attention, resulting in a total cost of $\mathcal{O}(N+E)$ in terms of nodewise communication operations. Note that although we are highlighting the complexity of graph convolution compared to that of linear attention, $\mathcal{O}(N)$, given our architecture also has other components that act pointwise such as linear layers (like all architectures in general) it is more accurate to think of the overall complexity of SMPNNs as being $\mathcal{O}(N+E)$. Unlike other models, we do not apply $\mathcal{O}(N^3)$ pre-processing steps (Dwivedi & Bresson, 2021; Ying et al., 2021; Chen et al., 2022; Hussain et al., 2022; Rampášek et al., 2022), which would be prohibitively expensive for the large graph datasets we are targeting. Additionally, our model does not require positional encodings, attention, augmented training loss, or edge embeddings to achieve competitive performance. We would like to highlight that the majority of Graph Transformers in Table 1 have been designed for smaller graphs and have only been demonstrated on datasets with thousands of nodes or fewer due to their quadratic complexity. Our main competitors are therefore NodeFormer (Wu et al., 2022), DIFFormer (Wu et al., 2023a), and SGFormer (Wu et al., 2023b), which have managed to scale to millions of nodes using less expressive or simplified versions of attention without quadratic complexity.

Table 1: Comparison of model components used by different Graph Transformers: Positional Encodings (PE), Multihead Attention (MA), Augmented Training Loss (ATL), and Edge Embeddings (EE). We also report whether the model uses All-Pair communication (typically implemented as some variant of attention), the Pre-processing and Training computational complexity, and the largest graph for which the method's performance has been reported. *Random regular expander graph generation is needed, and graphs that fail to be near-Ramanujan are discarded and regenerated.

Model	Model Components				All-Pair	Pre-processing	Training	Largest Demo
	PE	MA	AL	EE				
GraphTransformer (Dwivedi & Bresson, 2021)	✓	✓	✗	✓	✓	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$	0.2K
Graphormer (Ying et al., 2021)	✓	✓	✗	✓	✓	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$	0.3K
GraphTrans (Jain et al., 2021)	✗	✓	✗	✗	✓	-	$\mathcal{O}(N^2)$	0.3K
SAT (Chen et al., 2022)	✓	✓	✗	✗	✓	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$	0.2K
EGT (Hussain et al., 2022)	✓	✓	✓	✓	✓	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$	0.5K
GraphGPS (Rampášek et al., 2022)	✓	✓	✗	✓	✓	$\mathcal{O}(N^3)$	$\mathcal{O}(N+E)$	1.0K
Gophormer (Zhao et al., 2021)	✓	✓	✓	✗	✗	-	$\mathcal{O}(Nsm^2)$	20K
Exphormer (Shirzad et al., 2023)	✓	✓	✗	✓	✓	*	$\mathcal{O}(N+E)$	132K
NodeFormer (Wu et al., 2022)	✓	✓	✓	✗	✓	-	$\mathcal{O}(N+E)$	2.0M
DIFFormer (Wu et al., 2023a)	✗	✓	✗	✗	✓	-	$\mathcal{O}(N+E)$	1.6M
SGFormer (Wu et al., 2023b)	✗	✗	✗	✗	✓	-	$\mathcal{O}(N+E)$	100M
SMPNN	✗	✗	✗	✗	✗	-	$\mathcal{O}(N+E)$	100M

4 Theoretical Justification

In this section, we theoretically justify our architectural design. First, we review some theoretical findings from the literature that hint at the importance of residual connections to avoid oversmoothing in message-passing convolutions over graphs; moreover, by the results of Riedi et al. (2023) they make the loss landscape of the deep learning model much less jagged. Inspired by these results, we further extend the analysis through the lenses of universal approximation.

4.1 Oversmoothing

Oversmoothing is regarded as the tendency of node features to approach the same value after several message-passing transformations. This phenomenon has prompted the adoption of relatively shallow

GNNs, as adding many layers has traditionally resulted in node-level features that are too similar to each other and, hence, indistinguishable for downstream learners. This has hindered scalability and led to models with orders of magnitude fewer parameters than, for instance, counterparts in the literature on 2D generative modelling (Esser et al., 2024) and LLMs (Touvron et al., 2023), which have already adopted architectures with billions of parameters.

In general, standalone message-passing GNNs tend to behave as low-pass filters and may consequently lead to oversmoothing. However, this does not always need to be the case: graph convolution can also magnify high frequencies if it is augmented with residual connections. Following (Giovanni et al., 2023), message-passing is characterized as being *low- frequency dominant* (LFD) or *high-frequency dominant* (HFD) depending on the asymptotic behavior of the normalized Dirichlet energy of the system as the number of diffusion layers $l \to \infty$. These lead to oversmoothing and oversharpening, respectively.

Definition 4.1 (Graph Dirichlet Energy of a Message Passing System (Zhou & Scholkopf, 2005))**.** *We define the Dirichlet Energy as a map* $\mathfrak{E}^{Dir} : \mathbb{R}^{N\times D} \to \mathbb{R}$

$$\mathfrak{E}^{Dir}(\mathbf{X}) \overset{\text{def.}}{=} \frac{1}{2}\sum_{i,j\in\mathcal{E}} ||(\nabla\mathbf{X})_{ij}||^2 \,;\, (\nabla\mathbf{X})_{ij} \overset{\text{def.}}{=} \frac{\mathbf{x}_j}{\sqrt{\deg(v_j)}} - \frac{\mathbf{x}_i}{\sqrt{\deg(v_i)}}, \tag{9}$$

where $(\nabla\mathbf{X})_{ij}$ *is the edge-wise gradient of the graph node features* $\mathbf{X} \in \mathbb{R}^{N\times D}$.

Definition 4.2 (Graph Frequency Dominance (Giovanni et al., 2023))**.** *We say a message passing GNN is LFD if its normalized Dirichlet energy* $\mathfrak{E}^{Dir}(\mathbf{X}^{(l)})/||\mathbf{X}^{(l)}||^2 \to \lambda_1 = 0$ *for* $l \to \infty$ *and HFD if* $\mathfrak{E}^{Dir}(\mathbf{X}^{(l)})/||\mathbf{X}^{(l)}||^2 \to \lambda_n$ *for* $l \to \infty$*, where* $\mathbf{X}^{(l)}$ *encapsulates the node level features at a given layer* l *and* $1 \leqslant \lambda_n < 2$ *(Chung, 1996) is the largest eigenvalue whose corresponding eigenvector of the normalized graph Laplacian captures fine-grained microscopic behavior of the signal on the graph.*

In particular, the above characterization relies on analyzing message-passing from the perspective of gradient flows in the limit of infinitely many layers. Although analyzing the asymptotic behavior of GNNs is interesting from a theoretical perspective, the normalized Dirichlet energy is not directly predictive of model success, since both over-smoothing and over-sharpening will lead to degradation in performance (Giovanni et al., 2023). We provide an alternative perspective through the lens of universal approximation, which does not rely on asymptotic behavior. Specifically, we demonstrate that the universal approximation property of downstream classifiers is compromised when passing input features through graph convolution layers, but it can be restored with a residual connection.

4.2 Universality and Residual Connections

We will use $\mathcal{C}(\mathbb{R}^{N\times D})$ to denote the set of continuous functions from $\mathbb{R}^{N\times D}$ to $\mathbb{R}$, when both are equipped with the (only) norm topologies thereon. We use $\mathcal{MLP}_{N\times D}$ to denote the set of MLPs from $\mathbb{R}^{N\times D}$ to $\mathbb{R}$ with a SiLU activation function. We consider a class of models to be expressive if it is a universal approximator, in the local uniform sense of Hornik et al. (1989); Kidger & Lyons (2020); Duan et al. (2023); Yarotsky (2024), meaning that it can approximately implement any continuous function uniformly on compact sets.

Definition 4.3 (Universal Approximator)**.** *A subset of models* $\mathcal{F} \subseteq \mathcal{C}(\mathbb{R}^{N\times D})$ *is said to be a universal approximator in* $\mathcal{C}(\mathbb{R}^{N\times D})$ *if: for each continuous target function* $f : \mathbb{R}^{N\times D} \to \mathbb{R}$*, each uniform approximation error* $\varepsilon > 0$ *and every non-empty compact set of inputs* $K \subset \mathbb{R}^{N\times D}$*, there exists some model* $\hat{f} \in \mathcal{F}$ *such that*

$$\max_{\mathbf{X}\in K} |f(\mathbf{X}) - \hat{f}(\mathbf{X})| < \varepsilon. \tag{10}$$

We will consider two classes of models, highlighting the key benefits and drawbacks of adding versus omitting residual connections in graph convolution layers (Equation 2). Fix a matrix $\mathbf{W} \in \mathbb{R}^{D\times D}$ and a connected graph $\mathcal{G}$ on N nodes.

No Residual Connection. Consider the class $\mathcal{F}_{\mathcal{G},\mathbf{W}}$ which consists of all maps $\hat{f} : \mathbb{R}^{N\times D} \to \mathbb{R}$

$$\hat{f} = \text{MLP} \circ \mathcal{L}^{\text{conv}}_{\mathcal{G},\mathbf{W}} \,;\, \mathcal{L}^{\text{conv}}_{\mathcal{G},\mathbf{W}}(\mathbf{X}) \overset{\text{def.}}{=} \tilde{\mathbf{A}}\mathbf{X}\mathbf{W}, \tag{11}$$

where $\mathrm{MLP} \in \mathcal{MLP}_{N\times D}$ and $\tilde{\mathbf{A}}$ is the degree-normalized adjacency matrix. The class $\mathcal{F}_{\mathcal{G},\mathbf{W}}$ is the stylized representation of GCN models without a residual connection.

Residual Connection. We benchmark this class against a simplified version of our SMPNN block. Again, fixing $\mathbf{W}$ and a connected graph $\mathcal{G}$ on N nodes. This class, denoted by $\mathcal{F}^{\text{residual}}_{\mathcal{G},\mathbf{W}}$, consists of maps $\hat{f} : \mathbb{R}^{N\times D} \to \mathbb{R}$ with representation

$$\hat{f} = \mathrm{MLP} \circ \mathcal{L}^{\text{conv+r}}_{\mathcal{G},\mathbf{W}} \,;\, \mathcal{L}^{\text{conv+r}}_{\mathcal{G},\mathbf{W}}(\mathbf{X}) \overset{\text{def.}}{=} \tilde{\mathbf{A}}\mathbf{X}\mathbf{W} + \mathbf{X}, \tag{12}$$

We emphasize that the MLPs within both sets, $\mathcal{F}_{\mathcal{G},\mathbf{W}}$ and $\mathcal{F}^{\text{residual}}_{\mathcal{G},\mathbf{W}}$, are identical. Hence, there are no inherent advantages or disadvantages between them; the sole discrepancy lies in the presence or absence of a residual connection. The following set of results provides a counter-example to the universality of $\mathcal{F}_{G,\mathbf{W}}$ when G is a complete graph. We shows that this deficit it not an issue when incorporating a residual connection as $\mathcal{F}^{\text{residual}}_{G,\mathbf{W}}$ is universal.

In general, analyzing the expressive power of GNNs from the perspective of universal approximation can be cumbersome due to the variety of vastly different graphs encountered in practice. However, the importance of accompanying graph convolution with a residual connection can already be observed in the case of a complete graph. Our first main finding is a *negative result*, showing that the universal approximation property of MLPs is lost when passing input features through a graph convolution layer with no residual connection for *any* learnable weight matrix $\mathbf{W}$ used in equation 11.

Theorem 4.4 (No Universal Approximation via Graph Convolution Alone)**.** *Let N, D be positive integers with $N \geqslant 2$. Let $\mathcal{G}$ be a complete graph on N nodes. For any weight matrix $\mathbf{W} \in \mathbb{R}^{D\times D}$, the class $\mathcal{F}_{\mathcal{G},\mathbf{W}}$ defined in equation 11 is not a universal approximator in $\mathcal{C}(\mathbb{R}^{N\times D})$.*

This is contrasted against our *positive result*, which shows that the universal approximation property of MLPs is preserved when passing input features through graph convolution with a residual connection.

Theorem 4.5 (Universal Approximation via Graph Convolution is Possible with Residual Connections)**.** *Let N, D be positive integers. There exists a weight matrix $\mathbf{W} \in \mathbb{R}^{D\times D}$ such that the class "with skip connection" $\mathcal{F}^{\text{residual}}_{\mathcal{G},\mathbf{W}}$ is a universal approximator in $\mathcal{C}(\mathbb{R}^{N\times D})$.*

Interestingly, a more technical version of Theorem 4.5 in the appendix of the paper (Theorem B.6), which shows that the matrix $\mathbf{W}$ allows for universality to be regained through the skip connection is not obscure; rather it can be explicitly constructed by sampling from a random Gaussian matrix. Furthermore such a matrix is nearly always obtained in this manner with probability approaching 1 in high dimensions. All proofs and derivations are provided in Appendix B.

This analysis supports our architectural choices in Section 3.2, which represent an amalgamation of best practices in the LLM literature with the theoretical insights here presented concerning oversmoothing and universality.

5 Experimental Validation

We experimentally validate our architecture and compare it to recent state-of-the-art baselines. We follow the evaluation protocol used in NodeFormer (Wu et al., 2022), DIFFormer (Wu et al., 2023a), and SGFormer (Wu et al., 2023b). Additionally, we perform ablations and further experiments to verify the theoretical analysis presented in Section 4. For details, refer to Appendix C.

Large Scale Graph Datasets (Table 2). We compare the performance of SMPNNs to that of Graph Transformers tailored for large graph transductive learning, as well as other GNN baselines. Note that most Graph Transformer architectures (Table 1) are difficult to scale to these datasets. We observe that SMPNNs consistently outperform SOTA architectures without the need for attention. Augmenting the base SMPNN model with linear attention (Appendix A) leads to improvements in performance of under 1% only, while substantially increasing computational overhead. For instance, in the case of the ogbn-products dataset, using the same hyperparameters, adding linear global attention with a single head increases the total number of model parameters from 834K to 2.4M. The resulting performance gain of only 0.18% requires thus more than twice as many parameters.

Table 2: Test results (mean ± standard deviation over 5 runs) on large graph datasets. We report ROC-AUC for ogbn-proteins and accuracy for all others, higher is better.

Dataset	**ogbn-proteins**	**pokec**	**ogbn-arxiv**	**ogbn-products**
Nodes	132,534	1,632,803	169,343	2,449,029
Edges	39,561,252	30,622,564	1,166,243	61,859,140
MLP	72.04 ± 0.48	60.15 ± 0.03	55.50 ± 0.23	63.46 ± 0.10
GCN (Kipf & Welling, 2017b)	72.51 ± 0.35	62.31 ± 1.13	71.74 ± 0.29	83.90 ± 0.10
SGC (Wu et al., 2019)	70.31 ± 0.23	52.03 ± 0.84	67.79 ± 0.27	81.21 ± 0.12
GCN-NSampler Kipf & Welling (2017b); Zeng et al. (2020)	73.51 ± 1.31	63.75 ± 0.77	68.50 ± 0.23	83.84 ± 0.42
GAT-NSampler Veličković et al. (2018b); Zeng et al. (2020)	74.63 ± 1.24	62.32 ± 0.65	67.63 ± 0.23	85.17 ± 0.32
SIGN (Frasca et al., 2020)	71.24 ± 0.46	68.01 ± 0.25	70.28 ± 0.25	80.98 ± 0.31
NodeFormer (Wu et al., 2022)	77.45 ± 1.15	68.32 ± 0.45	59.90 ± 0.42	87.85 ± 0.24
Difformer (Wu et al., 2023a)	79.49 ± 0.44	69.24 ± 0.76	64.94 ± 0.25	84.00 ± 0.07
SGFormer (Wu et al., 2023b)	**79.53 ± 0.38**	**73.76 ± 0.24**	**72.63 ± 0.13**	**89.09 ± 0.10**
SMPNN	**83.15 ± 0.24**	**79.76 ± 0.19**	**73.75 ± 0.24**	**90.61 ± 0.05**
SMPNN + Attention	**83.65 ± 0.35**	**80.09 ± 0.12**	**74.38 ± 0.16**	**90.79 ± 0.04**
Linear Transformer	60.87 ± 7.23	62.72 ± 0.09	58.02 ± 0.21	79.85 ± 0.08

Most Graph Transformers in the literature use multiple attention heads (Table 1); we experimented with up to 4 attention heads but found differences in terms of performance not to be statistically significant, which seems to align with the conclusions of (Wu et al., 2023b). Furthermore, note that as reported in the OGB benchmark paper (Hu et al., 2020), the Max Strongly Connected Component Ratio (MaxSCC Ratio) for the node-level datasets used in these experiments is 1.00 for all datasets (except for ogbn-products, which is 0.97). This indicates that the entire graph is a single strongly connected component, meaning every node is reachable from every other node. Although this does not guarantee that information bottlenecks will not be present, a high MaxSCC Ratio denotes high inter-connectivity, where information or influence can rapidly spread among a large proportion of the network, which may explain why attention is not that important in such settings. Apart from SMPNN and SMPNN augmented with attention, we also include a Linear Transformer baseline, which corresponds to removing the GCN layer from our architecture and substituting local convolution with global linear attention. This is also clearly worse than SMPNNs, which again verifies that in these datasets, the locality inductive bias is important.

Table 3: Test results (mean accuracy ± standard deviation) on ogbn-papers-100M dataset.

Dataset	**ogbn-papers-100M**
Nodes	111,059,956
Edges	1,615,685,872
MLP	47.24 ± 0.31
GCN	63.29 ± 0.19
SGC	63.29 ± 0.19
GCN-NSampler	62.04 ± 0.27
GAT-NSampler	63.47 ± 0.39
SIGN	**65.11 ± 0.14**
SGFormer	**66.01 ± 0.37**
SMPNN	**66.21± 0.10**

100M Node Graph Dataset and Scalability Experiments (Table 3). We test the scalability of our pipeline on the ogbn-papers-100M dataset. Our main competitors are SIGN (Frasca et al., 2020) and SGFormer (Wu et al., 2023b), as other Graph Transformers have not been able to scale to this dataset. We outperform SGFormer without requiring attention, again demonstrating the scalability and effectiveness of SMPNNs.

Table 4: Test results (mean MSE ± standard deviation over 5 runs) for spatio-temporal dynamics prediction datasets. Lower is better. *w/o g* stands for *without graph*.

Dataset	**Chickenpox**	**Covid**	**WikiMath**
MLP	0.924 ± 0.001	0.956 ± 0.198	1.073 ± 0.042
GCN	0.923 ± 0.001	1.080 ± 0.162	1.292 ± 0.125
GAT	0.924 ± 0.002	1.052 ± 0.336	1.339 ± 0.073
GCN-kNN	0.936 ± 0.004	1.475 ± 0.560	1.023 ± 0.058
GAT-kNN	0.926 ± 0.004	0.861 ± 0.123	0.882 ± 0.015
DenseGAT	0.935 ± 0.005	1.524 ± 0.319	0.826 ± 0.070
DIFFormer-s	**0.914 ± 0.006**	0.779 ± 0.037	0.731 ± 0.007
DIFFormer-a	**0.915 ± 0.008**	**0.757 ± 0.048**	0.763 ± 0.020
DIFFormer-s w/o g	**0.916 ± 0.006**	0.779 ± 0.028	**0.727 ± 0.025**
DIFFormer-a w/o g	**0.916 ± 0.006**	**0.741 ± 0.052**	**0.716 ± 0.030**
SMPNN	**0.916 ± 0.006**	**0.756 ± 0.048**	**0.713 ± 0.032**

Additional Image, Text, and Spatio-Temporal Benchmarks (Tables 5 and 4). For completeness, we also test our architecture on image and text classification with low label rates, as well as on spatio-temporal dynamics prediction, following (Wu et al., 2023a). Our model achieves the best performance in most configurations for the CIFAR, STL, and 20News datasets. Likewise, in Table 4, SMPNNs are also competitive in spatio-temporal prediction. These results demonstrate that our architecture is applicable to a variety of tasks.

Table 5: Test results (mean accuracy ± standard deviation over 5 runs) on Image and Text datasets.

Dataset	CIFAR			STL			20News		
Labels	100	500	1000	100	500	1000	1000	2000	4000
MLP	65.9 ± 1.3	73.2 ± 0.4	75.4 ± 0.6	66.2 ± 1.4	**73.0 ± 0.8**	75.0 ± 0.8	54.1 ± 0.9	57.8 ± 0.9	62.4 ± 0.6
ManiReg	67.0 ± 1.9	72.6 ± 1.2	74.3 ± 0.4	66.5 ± 1.9	72.5 ± 0.5	74.2 ± 0.5	56.3 ± 1.2	60.0 ± 0.8	63.6 ± 0.7
GCN-kNN	66.7 ± 1.5	72.9 ± 0.4	74.7 ± 0.5	**66.9 ± 0.5**	72.1 ± 0.8	73.7 ± 0.4	56.1 ± 0.6	60.6 ± 1.3	64.3 ± 1.0
GAT-kNN	66.0 ± 2.1	72.4 ± 0.5	74.1 ± 0.5	66.5 ± 0.8	72.0 ± 0.8	73.9 ± 0.6	55.2 ± 0.8	59.1 ± 2.2	62.9 ± 0.7
Dense-GAT	OOM	OOM	OOM	OOM	OOM	OOM	54.6 ± 0.2	59.3 ± 1.4	62.4 ± 1.0
GLCN	66.6 ± 1.4	72.8 ± 0.5	74.7 ± 0.3	66.4 ± 0.8	72.4 ± 1.3	74.3 ± 0.7	56.2 ± 0.8	60.2 ± 0.7	64.1 ± 0.8
Differmer-a	**69.3 ± 1.4**	**74.0 ± 0.6**	**75.9 ± 0.3**	66.8 ± 1.1	72.9 ± 0.7	**75.3 ± 0.6**	**57.9 ± 0.7**	**61.3 ± 1.0**	**64.8 ± 1.0**
Differmer-s	**69.1 ± 1.1**	**74.8 ± 0.5**	**76.6 ± 0.3**	**67.8 ± 1.1**	**73.7 ± 0.6**	**76.4 ± 0.5**	57.7 ± 0.3	61.2 ± 0.6	**65.9 ± 0.8**
SMPNN	**68.6 ± 1.8**	**76.2 ± 0.5**	**78.0 ± 0.3**	**67.9 ± 0.9**	**73.9 ± 0.7**	**76.7 ± 0.5**	**58.9 ± 0.8**	**62.7 ± 0.6**	**65.6 ± 0.6**

Ablation on SMPNN Architecture Components. In Table 6, we conduct an ablation study. First, we ablate the standard SMPNN architecture by removing residual connections after graph convolution layers. In line with the theory presented in Section 4, we observe that this indeed has a significant impact on model performance. Next, we test the effect of fixing the learnable scaling connections ($\alpha_1^{(l)} = \alpha_2^{(l)} = 1, \forall l$) while retaining the residuals. This decreases performance for ogbn-proteins and ogbn-arxiv, whereas it slightly increases performance for pokec and ogbn-products. We decide to keep the scalings since they make the model more expressive in general. Additionally, we test removing the pointwise feedforward transformation, which leads to a slight drop in performance for all datasets and seems particularly relevant for ogbn-proteins. This experiment suggests that the most critical part of the architecture is the message-passing component, as expected. Lastly, for completeness, we also test removing the LayerNorm before the GCN, which generally leads to a drop in performance. However, in the case of ogbn-arxiv, this configuration obtains a test accuracy of 74.46%, which outperforms all other models and baselines. In conclusion, there is no free lunch; depending on the dataset, slight modifications may lead to improved performance and, interestingly, have even more of an effect than augmenting SMPNNs with linear attention. Nevertheless, we find that, in general, the standard SMPNN block is robust and leads to SOTA results across several datasets.

Deep Models are Possible with SMPNNs. It is well known that conventional message-passing GNNs (Kipf & Welling, 2017b; Veličković et al., 2018a; Brody et al., 2021) are restricted to shallow architectures, since their performance degrades when stacking many layers. Next, we demonstrate that deep models are possible with SMPNNs. In particular, we perform an experiment in which we progressively increase the number of SMPNN layers while keeping all other hyperparameters constant for the ogbn-arxiv and ogbn-proteins datasets, from 2 to 12 layers. Adding up to 6 layers seems to improve performance, and it plateaus thereafter. Additionally, we perform another experiment in which we follow the same procedure but for an SMPNN without residual connections after convolutions. In this case, we observe a clear drop in performance after 4 layers, which aligns with our theoretical understanding from Section 4. In the case of classical GCNs we achieve best performance at 4 layers with a test set accuracy of 72 ± 1.20 and at 8 layers the performance already drops to 42.86 ± 28.03.

Table 6: Results (mean test set accuracy ± standard deviation over 5 runs) for ablation studies on OGBN large graph datasets.

Model	Removed	ogbn-proteins	pokec	ogbn-arxiv	ogbn-products
SGFormer	N/A	79.53 ± 0.38	73.76 ± 0.24	72.63 ± 0.13	89.09 ± 0.10
SMPNN	N/A	**83.15 ± 0.24**	**79.76 ± 0.19**	**73.75 ± 0.24**	**90.61 ± 0.05**
SMPNN	Residual	68.49 ± 2.59	68.17 ± 8.22	39.67 ± 14.60	89.69 ± 0.05
SMPNN	α	**82.90 ± 0.34**	**80.10 ± 0.12**	73.00 ± 0.59	**90.77 ± 0.04**
SMPNN	FF	80.51 ± 0.61	78.40 ± 0.20	**73.25 ± 0.58**	90.01 ± 0.10
SMPNN	GCN LNorm	**80.74 ± 0.91**	**79.42 ± 0.15**	**74.46 ± 0.22**	**90.54 ± 0.08**

Table 7: Results (mean accuracy ± standard deviation over 5 runs) on deep SMPNN architectures.

		ogbn-arxiv, No. layers						
Model	Removed	2	3	4	6	8	10	12
SMPNN	N/A	73.18 ± 0.30	73.33 ± 0.38	73.65± 0.49	73.75 ± 0.24	73.71 ± 0.54	73.68 ± 0.51	73.73 ± 0.50
SMPNN	Residual	72.56 ± 0.91	72.61 ± 0.83	71.38 ± 1.48	39.67 ± 14.60	25.87 ± 1.43	26.10 ± 1.68	26.59 ± 2.25
		ogbn-proteins, No. layers						
Model	Removed	2	3	4	6	8	10	12
SMPNN	N/A	81.44 ± 0.36	82.57 ± 0.45	82.64 ± 0.73	83.15 ± 0.24	83.36 ± 0.34	83.28 ± 0.36	83.45 ± 0.41
SMPNN	Residual	76.83 ± 1.56	68.93 ± 2.41	69.98 ± 3.08	68.49 ± 2.59	64.22 ± 1.22	65.54 ± 0.99	65.90 ± 1.35

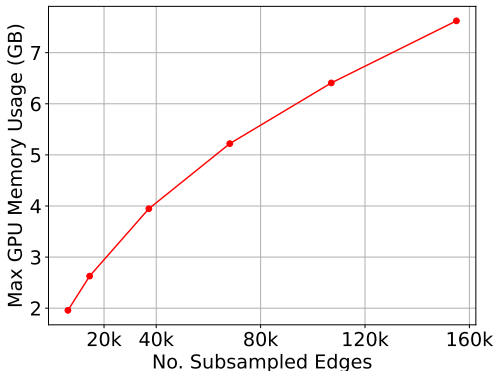

Figure 2: Max GPU consumption versus number of edges in the subgraph for SMPNN with 6 layers.

GPU Memory Scaling. We proceed to examine the model's scalability utilizing the ogbn-products dataset and randomly sampling a subset of nodes for graph mini-batching (from 10K to 100K), following the analysis in (Wu et al., 2023b). In Figure 2, we report the maximum GPU memory usage in GB against the number of edges in the subgraph induced by the sampled nodes. We can see that it asymptotes towards linearity in the number of edges as discussed in Section 3.3. For small subgraphs N dominates the complexity term and results in a steeper slope. Additionally, we compare the computational overhead of different SMPNN configurations to that of SGFormer on a Tesla V100 GPU, see Figure 3. Interestingly, the scaling factors α seem to have a substantial computational impact as the number of nodes increases, which, combined with the ablations in Table 6, suggests it may be better to remove it for very large graphs. Moreover, we find that the GPU consumption of SMPNNs can be substantially reduced when omitting the pointwise feedforward part of the block. This reduces computation cost substantially below that of SGFormer. Hence, one can trade GPU usage for performance: note that as shown in Table 6, SMPNNs without feedforwards still outperform the SGFormer baseline. SGFormer does not use feedforward networks or traditional attention mechanisms. Instead, it employs a simplified linear version of attention and even omits the softmax activation function.

6 Conclusion

Our framework, Scalable Message Passing Neural Networks (SMPNNs), overcomes scalability issues in traditional GNNs. SMPNNs scale efficiently to large graphs, outperforming Graph Transformers without using global attention. They also allow for deep architectures without degradation in performance, a common limitation of GNNs. Empirical validation across diverse datasets confirms SMPNNs' competitive performance and validates the theoretical justification of residual connections. The present work opens new possibilities for leveraging deep architectures in large-scale graph learning.

Ethical Statement Our model enhances deep learning for large graphs and networks, which could help social media companies track user behaviour. Since methods for large-scale machine learning on graphs already exist, our improvements would only slightly increase existing risks.

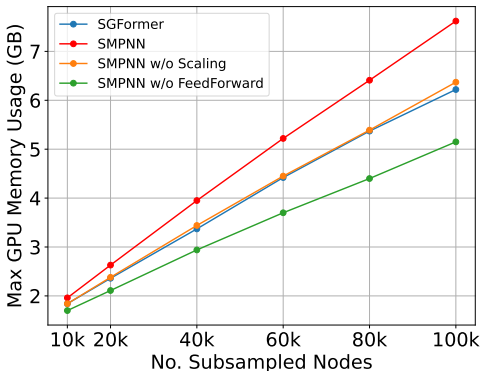

Figure 3: Max GPU consumption versus the number of nodes in the batch subgraph for different models with 6 layers.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling, 2019.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks?, 2021.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, 2022.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019.

Fan R. K. Chung. Spectral graph theory. 1996. URL https://api.semanticscholar.org/CorpusID:60624922.

Gbètondji J-S Dovonon, Michael M. Bronstein, and Matt J. Kusner. Setting the record straight on transformer oversmoothing, 2024.

Yifei Duan, Guanghua Ji, Yongqiang Cai, et al. Minimum width of leaky-relu neural networks for uniform universal approximation. In *International Conference on Machine Learning*, pp. 19460–19470. PMLR, 2023.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis, 2024.

Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Benjamin Chamberlain, Michael Bronstein, and Federico Monti. Sign: Scalable inception graph neural networks. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*, 2020.

Francesco Di Giovanni, James Rowbottom, Benjamin Paul Chamberlain, Thomas Markovich, and Michael M. Bronstein. Understanding convolution on graphs via energies. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=v5ew3FPTgb.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Cambridge University Press, Cambridge, second edition, 2013. ISBN 978-0-521-54823-6.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

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

Md Shamim Hussain, Mohammed J. Zaki, and Dharmashankar Subramanian. Global self-attention as a replacement for graph convolution. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pp. 655–665, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539296. URL https://doi.org/10.1145/3534678.3539296.

Paras Jain, Zhanghao Wu, Matthew A. Wright, Azalia Mirhoseini, Joseph E. Gonzalez, and Ion Stoica. Representing long-range context for graph neural networks with global attention. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=nYz2_BbZnYk.

Patrick Kidger and Terry Lyons. Universal approximation with deep narrow networks. In *Conference on learning theory*, pp. 2306–2327. PMLR, 2020.

Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, 2017a.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017b. URL https://openreview.net/forum?id=SJU4ayYgl.

Anastasis Kratsios and Ievgen Bilokopytov. Non-euclidean universal approximation. *Advances in Neural Information Processing Systems*, 33:10635–10646, 2020.

Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, October 2023.

Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195, 1999.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and D. Beaini. Recipe for a general, powerful, scalable graph transformer. *ArXiv*, abs/2205.12454, 2022. URL https://api.semanticscholar.org/CorpusID:249062808.

Rudolf H Riedi, Randall Balestriero, and Richard G Baraniuk. Singular value perturbation and deep network optimization. *Constructive Approximation*, 57(2):807–852, 2023.

Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Alexander Riedel, Maria Astefanoaei, Oliver Kiss, Ferenc Beres, Guzmán López, Nicolas Collignon, and Rik Sarkar. Pytorch geometric temporal: Spatiotemporal signal processing with neural machine learning models. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, pp. 4564–4573, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384469. doi: 10.1145/3459637.3482014. URL https://doi.org/10.1145/3459637.3482014.

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, 2009. doi: 10.1109/TNN.2008.2005605.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks, 2017.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*, 2023.

Noboru Suzuki. On the convergence of Neumann series in Banach space. *Math. Ann.*, 220(2):143–146, 1976. ISSN 0025-5831,1432-1807. doi: 10.1007/BF01351698. URL https://doi.org/10.1007/BF01351698.

Haitz Sáez de Ocáriz Borde. Elucidating graph neural networks, transformers, and graph transformers, 02 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017a. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017b. URL https://api.semanticscholar.org/CorpusID:13756489.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *ArXiv*, 2018a.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018b. URL https://openreview.net/forum?id=rJXMpikCZ.

Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. pp. 210–268, 2012.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. Learning deep transformer models for machine translation, 2019.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 6861–6871. PMLR, 2019.

Qitian Wu, Wentao Zhao, Zenan Li, David Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Qitian Wu, Chenxiao Yang, Wen-Long Zhao, Yixuan He, David Wipf, and Junchi Yan. Difformer: Scalable (graph) transformers induced by energy constrained diffusion. *International Conference on Learning Representations (ICLR)*, abs/2301.09474, 2023a. URL https://api.semanticscholar.org/CorpusID:256105170.

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Sgformer: Simplifying and empowering transformers for large-graph representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023b.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture, 2020. URL https://openreview.net/forum?id=B1x8anVFPr.

Dmitry Yarotsky. Structure of universal formulas. *Advances in Neural Information Processing Systems*, 36, 2024.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation?, 2021.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJe8pkHFwS.

Jianan Zhao, Chaozhuo Li, Qian Wen, Yiqi Wang, Yuming Liu, Hao Sun, Xing Xie, and Yanfang Ye. Gophormer: Ego-graph transformer for node classification. *ArXiv*, abs/2110.13094, 2021. URL https://api.semanticscholar.org/CorpusID:239768298.

Dengyong Zhou and Bernhard Scholkopf. Regularization on discrete spaces. In *DAGM-Symposium*, 2005. URL https://api.semanticscholar.org/CorpusID:16394993.

A AUGMENTING SMPNNS WITH ATTENTION

SMPNNs, in principle, perform local convolution but can also be enhanced with attention. Here, we discuss the specific attention mechanism implemented in our experiments.

Linear Global Attention. Global attention can be seen as computing message passing over a fully-connected dense graph. In our block, attention is computed with respect to a virtual node, instead of the full $\mathcal{O}(N^2)$ attention. Linear attention is more suitable for scaling to large graphs. Given the input query, key, and value matrices, $\mathbf{Q} = \mathbf{X}^{(l)}\mathbf{W}_Q \in \mathbb{R}^{N\times D}$, $\mathbf{K} = \mathbf{X}^{(l)}\mathbf{W}_K \in \mathbb{R}^{N\times D}$, $\mathbf{V} = \mathbf{X}^{(l)}\mathbf{W}_V \in \mathbb{R}^{N\times D}$ we apply the following operations. First, we compute a summed query value considering the contribution from all nodes, and normalize both the query and key matrices:

$$\mathbf{Q}_{\text{sn}} = \frac{\sum_{i=1}^{N}\mathbf{Q}_i}{||\sum_{i=1}^{N}\mathbf{Q}_i||_2} \in \mathbb{R}^D,\ \mathbf{K}_n = \frac{\mathbf{K}}{\|\mathbf{K}\|_2} \in \mathbb{R}^{N\times D}. \tag{13}$$

Based on these we can compute attention

$$\mathbf{A}_{attention} = \text{softmax}(\mathbf{Q}_{\text{sn}}\mathbf{K}_n^T) \in \mathbb{R}^{1\times N}, \tag{14}$$

and use it to aggregate the values

$$\mathbf{X}^{(l+1)} = (\mathbf{A}_{attention} \otimes \mathbf{1}_N)\,\mathbf{V} \in \mathbb{R}^{N\times D}, \tag{15}$$

where $\mathbf{1}_N$ is a row vector of ones with dimension $1 \times N$ and $\otimes$ is the Kronecker product. The operation $\mathbf{A}_{attention} \otimes \mathbf{1}_N$ results in and $N \times N$ matrix, with all rows sharing the same attention coefficients, which is later multiplied by the value matrix $\mathbf{V}$ to obtain the final updated node features $\mathbf{X}^{(l+1)}$. The operations above can be extended to encompass multi-head attention. In such case we would have a total of H attention heads and to compute the final global feature matrix we aggregate the contribution from all heads:

$$\mathbf{X}^{(l+1)} = \sum_{h=1}^{H}\left(\mathbf{A}^{(h)}_{attention} \otimes \mathbf{1}_N\right)\mathbf{V}^{(h)} \in \mathbb{R}^{N\times D}. \tag{16}$$

Message-Passing with Parallel Attention. To augment SMPNN we use the following procedure:

$$\mathbf{H}^{(l)}_{1,local} = \text{LayerNorm}(\mathbf{X}^{(l)}). \tag{17}$$

$$\mathbf{H}^{(l)}_{2,local} = \text{SiLU}\left(\tilde{\mathbf{A}}\mathbf{H}^{(l)}_{1,local}\mathbf{W}^{(l)}_1\right), \tag{18}$$

$$\mathbf{H}^{(l)}_{1,global} = \text{LayerNorm}(\mathbf{X}^{(l)}), \tag{19}$$

$$\mathbf{H}^{(l)}_{2,global} = \text{LinearGlobalAttention}\left(\mathbf{H}^{(l)}_{1,global}\right), \tag{20}$$

$$\mathbf{H}^{(l)}_{2} = \alpha^{(l)}_1\left(\mathbf{H}^{(l)}_{2,local} + \mathbf{H}^{(l)}_{2,global}\right) + \mathbf{X}^{(l)}, \tag{21}$$

where Layer Norms have different parameters for local and global features. Pointwise-feedforward operations are kept as in the original SMPNN formulation.

The update equations above are reminiscent of hybrid Graph Transformers (Rampášek et al., 2022; Wu et al., 2022; 2023a;b), and we find that they do not provide tangible improvements over standard SMPNNs. Note that we also experimented with other attention mechanisms, which did not help.

B PROOFS

This appendix contains all the proofs for our theoretical guarantees discussed in Section 4.2.

B.1 PROOFS FOR UNIVERSALITY

The proofs of both Theorems 4.4 and 4.5 revolve around verifying the conditions (or their failure) stated in (Kratsios & Bilokopytov, 2020, Theorem 3.4). This result characterizes situations where a continuous transformation of the inputs to a universal approximator preserves its universal approximation property. Importantly, when the output space is Euclidean (e.g., $\mathbb{R}$ in our case), it is both necessary and sufficient for the transformation applied to the input of the model class to be *injective*. Consequently, our proofs for both results primarily focus on the injectivity (or lack thereof) of the feature transformations in equation 11 and equation 12, respectively

In what follows, for any activation function $\zeta : \mathbb{R} \to \mathbb{R}$ we use $\mathcal{MLP}^{\zeta}_{N\times D}$ to denote the class of MLPs with activation function ζ, mapping $\mathbb{R}^{N\times D}$ to $\mathbb{R}$. Note that this is mildly more general than the class $\mathcal{MLP}_{N\times D}$ considered in our main text, which used the specification $\zeta = \mathrm{SiLU}$. We consider the following regularity condition on ζ.

Assumption B.1 (Regular Activation)**.** *The activation function $\zeta : \mathbb{R} \to \mathbb{R}$ is continuous and either:*

(i) Pinkus (1999): ζ is non-polynomial,

(ii) Kidger & Lyons (2020): ζ is non-affine and there exists some $t \in \mathbb{R}$ at which ζ is continuously differentiable and $\zeta'(t) \neq 0$.

B.1.1 NO UNIVERSALITY WITHOUT THE RESIDUAL CONNECTION

We first show that adding a single normalized graph convolution layer preceding an MLP is enough to negate the universal approximation property of the composite model in equation 11.

Lemma B.2 (Loss of Injectivity)**.** *Let N, D be positive integers with $N \geqslant 2$. Let $\mathcal{G}$ be a complete graph on N nodes, let $\tilde{\mathbf{A}} \overset{\text{def.}}{=} (I_{i\sim j \vee i=j}/\sqrt{\deg(v_i)\deg(v_j)})_{i,j=1}^{N}$, and let $\mathbf{W} \in \mathbb{R}^{D\times D}$. Then, the map*

$$\begin{aligned} \mathcal{L}^{\text{conv}}_{\mathcal{G},\mathbf{W}} : \mathbb{R}^{N\times D} &\to \mathbb{R}^{N\times D} \\ \mathbf{X} &\mapsto \tilde{\mathbf{A}}\mathbf{X}\mathbf{W} \end{aligned}$$

is not injective.

Proof of Lemma B.2. Since $\mathcal{G}$ is complete and $N \geqslant 1$, then $\tilde{A}_{i,j} = 1/N < \infty$ for each $i, j = 1, \ldots, N$. In particular, all its rows are identical. Therefore, for each $\mathbf{X} \in \mathbb{R}^{N\times D}$ all the rows of the product matrix $\tilde{\mathbf{A}}(\mathbf{X}\mathbf{W}) = (\tilde{\mathbf{A}}\mathbf{X}\mathbf{W})$ are identical too. Hence, $\mathcal{L}^{\text{conv}}_{\mathcal{G},\mathbf{W}}$ is not injective. □

The following is a more technical version of Theorem 4.4.

Lemma B.3 (No Universality without Residual Connection)**.** *Let ζ satisfy Assumption B.1. Let N, D be positive integers with $N \geqslant 2$. Let $\mathcal{G}$ be a complete graph on N nodes, $\mathbf{W} \in \mathbb{R}^{D\times D}$, and let $\mathcal{L}^{\text{conv}}_{\mathcal{G},\mathbf{W}}$ be as in Lemma B.2. The class of maps $f : \mathbb{R}^{N\times D} \to \mathbb{R}$ with representation*

$$f = \mathrm{MLP} \circ \mathcal{L}^{\text{conv}}_{\mathcal{G},\mathbf{W}}$$

where $\mathrm{MLP} : \mathbb{R}^{N\times D} \to \mathbb{R}$ is an MLP with activation function ζ, is not dense for the topology of uniform convergence on compacts sets in $\mathcal{C}(\mathbb{R}^{N\times D}, \mathbb{R})$.

Proof. By (Pinkus, 1999, Theorem 3.1) and (Kidger & Lyons, 2020, Theorem 3.2) the class $\mathcal{MLP}^{\zeta}_{N\times D}$ is dense in $\mathcal{C}(\mathbb{R}^{N\times D}, \mathbb{R})$ for the topology of uniform convergence on compacts sets since ζ was assumed

to be nonpolynomial[1]. Since $\mathcal{L}^{\text{conv}}_{\mathcal{G},\mathbf{W}}$ is continuous then (Kratsios & Bilokopytov, 2020, Proposition 3.7) implies that the class

$$\mathcal{MLP}^{\varsigma}_{N\times D} \circ \mathcal{L}^{\text{conv}}_{\mathcal{G},\mathbf{W}} \overset{\text{def.}}{=} \left\{ \text{MLP} \circ \mathcal{L}^{\text{conv}}_{\mathcal{G},\mathbf{W}} : \text{MLP} \in \mathcal{MLP}^{\varsigma}_{N\times D} \right\},$$

is dense in $\mathcal{C}(\mathbb{R}^{N\times D}, \mathbb{R})$ for the topology of uniform convergence on compacts sets *if and only if* $\mathcal{L}^{\text{conv}}_{\mathcal{G},\mathbf{W}}$ is injective. By Lemma equation B.2, this map is never injective whenever $N > 1$. Thus, $\mathcal{MLP}^{\varsigma}_{N\times D} \circ \mathcal{L}^{\text{conv}}_{\mathcal{G},\mathbf{W}}$ is not dense in $\mathcal{C}(\mathbb{R}^{N\times D}, \mathbb{R})$ for the topology of uniform convergence on compacts sets; i.e. $\mathcal{MLP}^{\varsigma}_{N\times D} \circ \mathcal{L}^{\text{conv}}_{\mathcal{G},\mathbf{W}}$ is not a universal approximator in the sense of Definition 4.3. □

Next, we show that adding the residual connection recovers the universal approximation property.

B.1.2 Universality with the residual connection

We will consider the case reflective of the networks at initialization, where the graph convolution layer with residual connection is initialized randomly. We consider weights with i.i.d. random Gaussian initializations with appropriately scaled variance.

Lemma B.4 (Injectivity Regained)**.** *Let N, D be positive integers. Fix a hyperparameter $\varsigma > 0$. Let $\mathcal{G}$ be a complete graph on N nodes, let $\tilde{\mathbf{A}} \overset{\text{def.}}{=} (I_{i\sim j \vee i=j}/\sqrt{\deg(v_i)\deg(v_j)})_{i,j=1}^{N}$, and let $\mathbf{W}$ be a $D\times D$ random matrix with i.i.d. entries $W_{i,j} \sim N(0,\varsigma^2)$. If $\varsigma < 1/(9D^{3/2})$ then*

$$\mathcal{L}^{\text{conv+r}}_{\mathcal{G},\mathbf{W}} : \mathbb{R}^{N\times D} \to \mathbb{R}^{N\times D}$$
$$\mathbf{X} \mapsto \tilde{\mathbf{A}}\mathbf{X}\mathbf{W} + \mathbf{X}$$

is injective with probability at-least $1 - e^{-D/2}$.

Our proof of the positive counterpart to Lemma B.3, namely Lemma B.4, will rely on the following result from random matrix theory.

Lemma B.5 (Concentration Version of Gordon's Theorem: Corollary 5.35 in Vershynin (2012))**.** *Let $\mathbf{A}$ be an $N\times n$ matrix whose entries are independent standard normal random variables. For every $t \geqslant 0$, the following holds*

$$\sqrt{N} - \sqrt{n} - t \leqslant s_{\min}(\mathbf{A}) \leqslant s_{\max}(\mathbf{A}) \leqslant \sqrt{N} + \sqrt{n} + t \tag{22}$$

with probability at least $1 - e^{-t^2/2}$; where $s_{\min}(\mathbf{A})$ and $s_{\max}(\mathbf{A})$ are respectively the smallest and largest singular values of the matrix $\mathbf{A}$.

Proof. **Step 1 - Reduction of injectivity problem to eigenvalue computation**
Since $\mathcal{L}^{\text{conv+r}}_{\mathcal{G},\mathbf{W}}$ is a linear map then the dimension theorem/rank-nullity theorem/splitting lemma implies that is injective if and only if its kernel only contains the zero vector; i.e.

$$\mathcal{L}^{\text{conv+r}}_{\mathcal{G},\mathbf{W}}(\mathbf{X}) = 0 \Leftrightarrow \mathbf{X} = 0. \tag{23}$$

Using the Kronecker product $\otimes$, see (Horn & Johnson, 2013, Section 4.2), and the "flattenning" vectorization map $\text{vec} : \mathbb{R}^{N\times D} \mapsto \mathbb{R}^{ND}$ which sends any $N\times D$ matrix to the vector obtained from ordering its entries in lexicographic order (according to their indices), see (Horn & Johnson, 2013, Definition 4.2.9), we may rewrite

$$(I_{ND} + \tilde{\mathbf{A}} \otimes \mathbf{W})\ \text{vec}(\mathbf{X}) = \mathcal{L}^{\text{conv+r}}_{\mathcal{G},\mathbf{W}}(\mathbf{X}) = 0 \Leftrightarrow \mathbf{X} = 0, \tag{24}$$

where I_{ND} is the $ND\times ND$-dimensional identity matrix. A necessary condition in equation 24 is that $I_{ND} + \tilde{\mathbf{A}} \otimes \mathbf{W}$ has to be invertible. Let $T \overset{\text{def.}}{=} -\tilde{\mathbf{A}} \otimes \mathbf{W}$; then, $I_{ND} + \tilde{\mathbf{A}} \otimes \mathbf{W} = I_{ND} - T$. The

[1]Note that this can be relaxed further since we are using deep, and not shallow, MLPs by (Kidger & Lyons, 2020, Theorem 3.2). However, this is not the main point of our analysis.

linear operator $I_{ND} - T$ is invertible [2] if $\|T\|_2 < 1$. Since $T = -\tilde{\mathbf{A}} \otimes \mathbf{W}$, $\| - T\|_2 = \|T\|_2$ then $(I_{ND} + \tilde{\mathbf{A}} \otimes \mathbf{W})$ is injective if

$$\|\tilde{\mathbf{A}} \otimes \mathbf{W}\|_2 < 1. \tag{25}$$

Recall that, $\|\cdot\|_2$ is the square-root of the sum of the squared eigenvalues of the matrix $\tilde{\mathbf{A}} \otimes \mathbf{W}$. By (Horn & Johnson, 2013, Theorem 4.2.12), the eigenvalues of $\tilde{\mathbf{A}} \otimes \mathbf{W}$ are $\{\lambda_i(\tilde{\mathbf{A}})\,\lambda_j(\mathbf{W})\}_{i=1,\dots,N,\,j=1,\dots,D}$ where $\lambda_i(\tilde{\mathbf{A}})$ (resp. $\lambda_j(\mathbf{W})$) is the i^{th} (resp. j^{th}) eigenvalue of $\tilde{\mathbf{A}}$ (resp. $\mathbf{W}$). As usual, we order its eigenvalues order in a non-increasing manner[3]. The Cauchy-Schwarz inequality and the definition of the norm $\|\cdot\|_2$ imply that

$$\|\tilde{\mathbf{A}} \otimes \mathbf{W}\|_2 \leqslant \|\tilde{\mathbf{A}}\|_2 \|\mathbf{W}\|_2 = \Big(\sum_{i=1}^{N} \lambda_i(\tilde{\mathbf{A}})^2\Big)^{1/2} \Big(\sum_{j=1}^{D} \lambda_j(\mathbf{W})^2\Big)^{1/2}. \tag{26}$$

Together, equation 25 and equation 26 provide the following sufficient condition for invertibility of the $I_{ND} + \tilde{\mathbf{A}} \otimes \mathbf{W}$; and thus for the injectivity of $\mathcal{L}^{\text{conv+r}}_{\mathcal{G},\mathbf{W}}$

$$\Big(\sum_{i=1}^{N} \lambda_i(\tilde{\mathbf{A}})^2\Big)^{1/2} \Big(\sum_{j=1}^{D} \lambda_j(\mathbf{W})^2\Big)^{1/2} < 1. \tag{27}$$

Step 2 - Computing the eigenvalues of $\tilde{\mathbf{A}}$

Since we have assumed that $\mathcal{G}$ is a complete graph on N nodes then $\tilde{\mathbf{A}}_{i,j} = 1/N$ for each $i, j = 1, \dots, N$. In particular, $\mathcal{G}$ has N identical rows. Thus, 0 is an eigenvalue of $\mathcal{G}$ with multiplicity at-least $N - 1$. Additionally, one easily verifies that $\mathbf{1}_N \overset{\text{def.}}{=} (1, \dots, 1) \in \mathbb{R}^N$ (the vector with all components equal to 1) is an eigenvector of $\tilde{\mathbf{A}}$. It corresponds to the eigenvalue $\lambda_1(\tilde{\mathbf{A}}) = 1$ since

$$\tilde{\mathbf{A}}\,\mathbf{1}_N = \lambda_1(\tilde{\mathbf{A}})\mathbf{1}_N = N\,\lambda_1(\tilde{\mathbf{A}}) = N\frac{1}{N} = 1.$$

Therefore,

$$\|\tilde{\mathbf{A}}\|_2 = \Big(\lambda_i(\tilde{\mathbf{A}})^2\Big)^{1/2} = \big(1^2 + (N-1)\,0^2\big)^{1/2} = 1. \tag{28}$$

Thus, the condition in equation 27 reduces further to

$$\Big(\sum_{j=1}^{D} \lambda_j(\mathbf{W})^2\Big)^{1/2} < 1. \tag{29}$$

It only remains to understand the "typical" eigenvalues of $\mathbf{W}$ (i.e. for most realization of the random matrix $\mathbf{W}$).

Step 3 - Computing the eigenvalues of $\mathbf{W}$

First, observe that

$$\Big(\sum_{j=1}^{D} \lambda_j(\mathbf{W})^2\Big)^{1/2} \leqslant \Big(\sum_{j=1}^{D} \lambda_1(\mathbf{W})^2\Big)^{1/2} \tag{30}$$

$$= \sqrt{D}|\lambda_1(\mathbf{W})| \tag{31}$$

$$= \sqrt{D}\, s_{\max}(\mathbf{W})^2. \tag{32}$$

Note that $W_{i,j} = \varsigma \tilde{W}_{i,j}$ where $\{\tilde{W}_{i,j}\}_{i,j=1}^{D}$ are i.i.d. standard normal random variables; meaning that the eigenvalues of $\mathbf{W}$ are exactly equal to ς times those of $\tilde{\mathbf{W}}$. By Gordon's Theorem, with $N = n = D$ and $t = \sqrt{D}$, the following holds with at-least $1 - e^{-D/2}$ probability:

$$\mathrm{s}_{\max}(\tilde{\mathbf{W}}) \leqslant 2\sqrt{D} + \sqrt{D} = 3\sqrt{D}. \tag{33}$$

[2] See (Suzuki, 1976, Theorem 1) for a complete characterization of the convergence of Neumann series, even in the infinite dimensional setting.

[3] Note that the order is not necessarily strictly increasing as eigenvalues may have non-trivial multiplicities.

Upon combining the inequalities in equation 32 and in equation 33 we deduce that: with at-least $1-e^{-D/2}$ probability the following holds

$$\Big(\sum_{j=1}^{D}\lambda_j(\mathbf{W})^2\Big)^{1/2} = \Big(\sum_{j=1}^{D}\varsigma^2\lambda_j(\tilde{\mathbf{W}})^2\Big)^{1/2} \leqslant \varsigma\sqrt{D}\,9D = \varsigma\,9\,D^{3/2}. \tag{34}$$

Note that equation 34 holds since the eigenvalues of $\mathbf{W}$ are equal to ς times the eigenvalues of $\tilde{\mathbf{W}}$ given that $\mathbf{W} = \varsigma\tilde{\mathbf{W}}$.

In particular, if $0 < \varsigma < 1/(9D^{3/2})$ then, with probability at-least $1 - e^{-D/2}$ we have that

$$\Big(\sum_{j=1}^{D}\lambda_j(\mathbf{W})^2\Big)^{1/2} \leqslant \varsigma\,9\,D^{3/2} < 1. \tag{35}$$

Step 4 - Conclusion
Consequentially, equation 35 implies that equation 29 is satisfied with probability at-least $1 - e^{-D/2}$; meaning that, the map $\mathcal{L}^{\text{conv+r}}_{\mathcal{G},\mathbf{W}}(\mathbf{X})$ is injective with probability at-least $1 - e^{-D/2}$ too. □

The following is a more technical version of Theorem 4.5; which shows that the result can be made to hold explicitly with high probability when appropriately sampling a weight matrix.

Theorem B.6 (Universality Regained)**.** *Suppose that ζ satisfies Assumption B.1. Let N, D be positive integers and fix a hyperparameter $0 < \varsigma < 1/(9D^{3/2})$. Let $\mathcal{G}$ be a complete graph on N nodes, let $\tilde{\mathbf{A}} \overset{\text{def.}}{=} (I_{i\sim j \vee i=j}/\sqrt{\deg(v_i)\deg(v_j)})_{i,j=1}^{N}$, let $\mathbf{W}$ be a $D \times D$ random matrix with i.i.d. entries $W_{i,j} \sim N(0,\varsigma^2)$, and define $\mathcal{L}^{\text{conv+r}}_{\mathcal{G},\mathbf{W}}$ as in Lemma B.4.*
Then, with probability at-least $1 - e^{-D/2}$, the class of maps $\hat{f} : \mathbb{R}^{N\times D} \to \mathbb{R}$ with representation

$$\hat{f} = \text{MLP} \circ \mathcal{L}^{\text{conv+r}}_{\mathcal{G},\mathbf{W}}$$

is dense in $\mathcal{C}(\mathbb{R}^{N\times D}, \mathbb{R})$ for the topology of uniform convergence on compacts sets, where $\text{MLP} : \mathbb{R}^{N\times D} \to \mathbb{R}$ is an MLP with activation function ζ.

Proof of Theorem B.6 (and thus of Theorem 4.5). By (Pinkus, 1999, Theorem 3.1) and (Kidger & Lyons, 2020, Theorem 3.2) the class $\mathcal{MLP}^{\zeta}_{N\times D}$ is dense in $\mathcal{C}(\mathbb{R}^{N\times D}, \mathbb{R})$ for the topology of uniform convergence on compacts sets since ζ satisfies Assumption B.1. By construction, the map $\mathcal{L}^{\text{conv+r}}_{\mathcal{G},\mathbf{W}}$ is continuous. By Lemma B.4 the map is injective with probability at-least $1 - e^{-D/2}$. Therefore, (Kratsios & Bilokopytov, 2020, Theorem 3.4) implies that the class

$$\mathcal{MLP}^{\zeta}_{N\times D} \circ \mathcal{L}^{\text{conv+r}}_{\mathcal{G},\mathbf{W}} \overset{\text{def.}}{=} \left\{ \text{MLP} \circ \mathcal{L}^{\text{conv+r}}_{\mathcal{G},\mathbf{W}} : \text{MLP} \in \mathcal{MLP}^{\zeta}_{N\times D} \right\}$$

is dense in $\mathcal{C}(\mathbb{R}^{N\times D}, \mathbb{R})$ for the topology of uniform convergence on compact sets. □

The probability appearing in Theorem 4.5 implies that, *at initialization*, there is a high probability $(1 - e^{-D/2})$ that the class $\mathcal{F}^{\text{residual}}_{\mathcal{G},\mathbf{W}}$ is expressive enough to learn any continuous function on $\mathcal{C}(\mathbb{R}^{N\times D})$ within large domains. Note that D is the hidden dimension, which tends to be high; thus, Theorem 4.5 holds with probability ≈ 1.

C Training Details

Dataset information. All datasets used in this work are publicly available. For OGB datasets, see (Hu et al., 2020); for Pokec, see (Leskovec & Krevl, 2014); and for the spatio-temporal datasets, see PyTorch Geometric Temporal (Rozemberczki et al., 2021). Following DIFFormer (Wu et al., 2023a), we perform

experiments on both image and text datasets to test the applicability of SMPNNs to a variety of tasks. The procedure described next is exactly the same as in (Wu et al., 2023a). We use STL(-10) and CIFAR(-10) as our image datasets. For the first, we use all 13,000 images, each classified into one of 10 categories. In the case of the latter, CIFAR, the dataset was originally pre-processed in previous work using 1,500 images from each of the 10 categories, resulting in a total of 15,000 images. For both datasets, 10, 50, or 100 instances per class were randomly selected for the training set (we use the same random splits as in the baselines for reproducibility and to obtain a fair comparison), 1,000 instances for validation, and the remaining instances for testing.

Train, validation, and test splits. We use all public data splits when available, such as in the case of the OGB benchmarks (Hu et al., 2020). Otherwise we follow the data split in the literature (Wu et al., 2022; 2023a;b) for a consistent comparison. Note that the code for these baselines is available online, including torch seeds for datasets with random splits. We use the *exact same* seeds (`123` in most cases).

Graph Sampling and Batching for Training and Inference. For ogbn-arxiv, CIFAR, STL, 20News, Chickenpox, Covid, and Wikipath, we employ full-graph training. Specifically, we input the entire graph into the model and simultaneously predict all node labels for the loss computation using a single GPU. During inference, we also use the entire graph as input and calculate the evaluation metric based on the predictions for the validation and test set nodes. For the rest of the datasets (except ogbn-papers-100M discussed later), due to their large sizes, we adopt the mini-batch training approach of (Wu et al., 2022; 2023a;b). More concretely, the subsampling procedure of the Graph Transformer baselines uses the following approach for training: a batch of nodes from the large graph dataset is selected at random, and based on this, the induced subgraph is sampled by extracting the edges connecting the selected nodes from the full graph adjacency matrix (the induced subgraph is obtained using `subgraph` from `torch geometric utils`). For inference, the entire graph is fed into the model on CPU. For the exceptionally large ogbn-papers100M graph, which cannot fit into CPU memory, we apply the same mini-batch partition strategy used during training for inference using a neighbor sampler with 3-hops and 15, 10, and 5 neighbors per hop using the standard `NeighborLoader` from `torch geometric`, in line with (Wu et al., 2023b).

GPU Memory Scaling Experiments. In these experiments, we use a single Tesla V100 GPU. We use the same training configuration for all networks: node batch sizes of 10K, 20K, 40K, 60K, 80K, and 100K (using `subgraph`), no weight decay, Adam optimizer with a learning rate of 10^{-3}, and dropout of 0.1. We set all model configurations to have 256 hidden channels and 6 layers, modifying only the parts of the configuration discussed in the main text. For SGFormer, we increase the number of GNN layers to 6 and fix the linear attention layer to 1, as in the original paper (Wu et al., 2023b).

Model Configurations. For most of the large-scale graph experiments (unless otherwise stated), we use SMPNNs with 6 layers. Note that we test other configurations; for instance, in Table 7. For the image, text, and spatio-temporal datasets, we use 2 layers to avoid overfitting. The exact command configurations can be found in the supplementary material.