Graph Inductive Biases in Transformers without Message Passing

Liheng Ma [*1 8] **Chen Lin** [*2] **Derek Lim** [3] **Adriana Romero-Soriano** [4 1 5 6] **Puneet K. Dokania** [2 7]
Mark Coates [1 8] **Philip H.S. Torr** [2] **Ser-Nam Lim** [4]

Abstract

Transformers for graph data are increasingly widely studied and successful in numerous learning tasks. Graph inductive biases are crucial for Graph Transformers, and previous works incorporate them using message-passing modules and/or positional encodings. However, Graph Transformers that use message-passing inherit known issues of message-passing, and differ significantly from Transformers used in other domains, thus making transfer of research advances more difficult. On the other hand, Graph Transformers without message-passing often perform poorly on smaller datasets, where inductive biases are more important. To bridge this gap, we propose the Graph Inductive bias Transformer (GRIT) — a new Graph Transformer that incorporates graph inductive biases without using message passing. GRIT is based on several architectural changes that are each theoretically and empirically justified, including: learned relative positional encodings initialized with random walk probabilities, a flexible attention mechanism that updates node and node-pair representations, and injection of degree information in each layer. We prove that GRIT is expressive — it can express shortest path distances and various graph propagation matrices. GRIT achieves state-of-the-art empirical performance across a variety of graph datasets, thus showing the power that Graph Transformers without message-passing can deliver.

[*]Equal contribution [1]McGill University [2]Department of Engineering Science, University of Oxford [3]CSAIL, Massachusetts Institute of Technology [4]MetaAI [5]Mila - Quebec AI Institute [6]Canada CIFAR AI Chair [7]FiveAI [8]International Laboratory on Learning Systems (ILLS). Correspondence to: Liheng Ma <liheng.ma@mail.mcgill.ca>, Chen Lin <chen.lin@eng.ox.ac.uk>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

1. Introduction

Following the success of Transformers (Vaswani et al., 2017) in different modalities like natural language processing (NLP) (Vaswani et al., 2017) and computer vision (Dosovitskiy et al., 2020), developing Transformers for graph data has attracted much interest (Dwivedi & Bresson, 2021; Kreuzer et al., 2021; Ying et al., 2021; Chen et al., 2022; Hussain et al., 2022; Rampášek et al., 2022; Zhang et al., 2023). A major motivation of Graph Transformers is to alleviate certain known limitations of (local) Message-Passing Graph Neural Networks (MPNNs) (Gilmer et al., 2017), such as over-smoothing (Li et al., 2018; Oono & Suzuki, 2020), over-squashing (Alon & Yahav, 2020; Topping et al., 2022), and expressive power limitations (Xu et al., 2019; Loukas, 2020; Morris et al., 2019).

However, it is known that transformers generally *lack strong inductive biases* (Dosovitskiy et al., 2020). In the graph domain, Graph Transformers aggregate information based on a learned attention matrix, which enjoys a high degree of freedom; in contrast, MPNNs explicitly aggregate the information according to the exact topology of the input graph. This comes with at least two consequences. First, Graph Transformers may be prone to over-fitting, and thus they often fail to outperform MPNNs in limited data settings. Second, learning meaningful attention scores typically requires capturing important positional or structural relationships between nodes, and strong positional encodings are challenging to design since the structure and symmetries of graph data are fundamentally different from that of other (Euclidean) domains (Vaswani et al., 2017; Bronstein et al., 2021). For instance, there is no ordering or canonical coordinate system for nodes in a graph, whereas words in a sentence have a sequence structure, and pixels in an image have a grid structure.

To incorporate graph inductive biases, many of the best-performing Graph Transformers explicitly integrate local message-passing mechanisms. For instance, some works incorporate sparse attention on local neighborhoods (Dwivedi & Bresson, 2021; Kreuzer et al., 2021), or integrate various other types of MPNN modules into their models (Chen et al., 2022; Rampášek et al., 2022). Thus, such Graph Transformers may at least partially inherit some of the lim-

itations of MPNNs. Moreover, message-passing modules within Graph Transformers make these models significantly different from the Transformers used in other domains; thus, it becomes more difficult to transfer some of the large quantities of Transformer research from other domains into the graph domain. This is exacerbated by the fact that integrating message-passing adds complexity to the design space of the underlying model, necessitating more architectural decisions and extra effort for hyperparameter tuning (Masters et al., 2022).

The trade-off between the limitations of message-passing and the importance of graph inductive biases can be seen in the empirical performance of models on competitive benchmarks. For instance, let us consider the results for two popular molecular graph regression benchmarks as of May 2023. On the small ZINC dataset (12,000 graphs) (Dwivedi et al., 2022a), GNNs that rely on message-passing take up the top spots on the leaderboards.[1] For the large PCQM4MV2 dataset (about 3,700,000 graphs) (Hu et al., 2021), Graph Transformers take up the top spots.[2]

In this work, we introduce the Graph Inductive bias Transformer (GRIT), a Graph Transformer that incorporates useful graph inductive biases without explicit message-passing modules. Our model is based on three design choices that integrate graph inductive biases, each of which is theoretically justified: (a) we use a learned relative positional encoding initialized with Relative Random Walk Probabilities (RRWP); this learned positional encoding can provably express shortest path distances (Ying et al., 2021) and general classes of message-passing propagations (Gasteiger et al., 2019; Zhao et al., 2021a; Xu et al., 2019). (b) we develop an attention mechanism that jointly updates both node representations and node-pair representations, and can thus learn to update the RRWP positional encodings; a distance based Weisfeiler Leman test (Zhang et al., 2023) shows that certain Graph Transformers with RRWP are strictly stronger than Graph Transformer with shortest path distances like Graphormer (Ying et al., 2021). (c) we inject degree information into our Transformer update using degree scalers, with batch normalization replacing the standard layer normalization; replacing the layer normalization is provably required to maintain the degree information.

Along with theoretical justification, we provide ample empirical evidence to demonstrate the effectiveness of our design choices. GRIT achieves state-of-the-art empirical performance across a variety of graph learning benchmarks, both small and large-scale. In particular, we bridge the performance gap in which message-passing-based methods do not perform as well in large datasets, and non-message-passing Transformers do not perform as well in small datasets. Ablations and synthetic experiments further justify our design decisions, showing that GRIT indeed integrates graph inductive biases into Transformers in an effective manner.[3]

[1] https://paperswithcode.com/sota/graph-regression-on-zinc-500k at the time of submission.

[2] https://ogb.stanford.edu/docs/lsc/leaderboards/

2. Related Work

Transformers for Euclidean Domains Transformers have achieved ground-breaking successes in various domains, including but not limited to natural language processing (Vaswani et al., 2017; Devlin et al., 2019; Dai et al., 2019) and computer vision (Dosovitskiy et al., 2020; Liu et al., 2021). As can be seen in both domains, Transformers often suffer from a lack of inductive bias compared to Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs), and typically require a large amount of training data in order to perform well. Recent studies have found that introducing more inductive biases can effectively improve the performance of Transformers (Liu et al., 2021; Park & Kim, 2021)

Positional Encoding and Structural Encoding for Graphs In the graph domain, positional encodings (PE) and structural encodings (SE) (Srinivasan & Ribeiro, 2020) have been studied for enhancing both Message-Passing Graph Neural Networks (MPNNs) and Graph Transformers (You et al., 2019; Ma et al., 2021; Li et al., 2020; Zhang et al., 2021; Dwivedi et al., 2022a; Loukas, 2020; Dwivedi et al., 2021; Lim et al., 2023; Wang et al., 2022). Such positional or structural encodings capture various types of graph features, such as shortest-path distances (Li et al., 2020), identity-awareness (You et al., 2021), and spectral information (Dwivedi et al., 2022a).

Many so-called positional or structural encodings contain both positional and structural information (Dwivedi et al., 2022a; Srinivasan & Ribeiro, 2020; Rampášek et al., 2022); thus, researchers use the terms positional encoding and structural encoding interchangeably in related literature. In this work, we use "positional encodings" as an umbrella term.

Graph Transformers with Message-Passing Dwivedi & Bresson (2021) is one of the early works that introduce a Transformer architecture to graph domains with Laplacian Positional Encoding (LapPE). However, the global attention version of it performs poorly compared to the message-passing-based sparse attention version. Kreuzer et al. (2021) propose the SAN model, which uses both sparse and global attention mechanisms at each layer, and introduces an extra

[3] The code and models are publicly available at https://github.com/LiamMa/GRIT.

transformer to encode LapPE. SignNet (Lim et al., 2023) is a specialized symmetry-invariant encoder for LapPE that uses message-passing modules to process Laplacian eigenvectors (Lim et al., 2023; Rampášek et al., 2022).

Several Graph Transformers (Chen et al., 2022; Rampášek et al., 2022) use random walk structural encodings (RWSE) (Dwivedi et al., 2021). Due in part to the fact that RWSE focuses on structural information and does not encode as much positional information, these Graph Transformers each integrate message-passing modules.

Graph Transformers without Message-Passing In contrast to the aforementioned works, a number of Graph Transformers without (local) message-passing have been proposed. Among them, a series of works propose to use relative positional encodings for each pair of nodes, such as shortest-path distance positional encodings (SPDPE) (Ying et al., 2021; Park et al., 2022; Luo et al., 2022). Generalized-Distance Transformers (Zhang et al., 2023) introduce other distances on graphs (e.g., resistance distances) as relative positional encodings. These works are among the most related to our method and typically perform well on large datasets such as the PCQM4Mv2 dataset from the OGB Large Scale Challenge (Hu et al., 2021). However, they perform worse on smaller datasets (e.g., ZINC (Dwivedi et al., 2022a)) compared to hybrid Graph Transformers and MPNNs (Rampášek et al., 2022). Based on a synthetic experiment (Sec. 4.3), we give evidence that due to the lack of sufficient inductive bias, they are not as capable in enabling attention mechanisms to perform local message-passing when necessary.

There are also several other Graph Transformers without message-passing proposed in recent years: TokenGT (Kim et al., 2022) proposes a Transformer that views both nodes and edges as tokens, and uses LapPE or orthogonal random features; Relational Transformer (Diao & Loynd, 2022) also proposes to update both node and edge tokens; EGT (Hussain et al., 2022) proposes to use an SVD-based PE instead of LapPE for directed graphs; (Mialon et al., 2021; Feldman et al., 2022) introduce positional encodings based on heat kernels and other graph kernels.

3. Methodology and Theory

In this section, we introduce our proposed GRIT architecture that uses a novel flexible attention mechanism together with a general relative positional encoding scheme, and does not incorporate any explicit local message-passing modules.

It is based on three design decisions: (i) a learned relative positional encoding initialized with random walk probabilities, (ii) a flexible attention mechanism that updates node and node-pair representations, and (iii) the integration of node degree information in each layer. Each of our design decisions is justified by theoretical results, which are also covered in this section. Further, in Section 4, we show that each of these design choices improves empirical performance in ablation studies, and the overall architecture achieves state-of-the-art performance across various datasets.

3.1. Learned Random Walk Relative Encodings

When applied to graph data, Transformers typically have a token embedding for each node and update this node embedding with attention and feedforward modules (FFNs). As the graph (adjacency) structure important to message-passing GNNs is removed from the architecture, the positional encodings must adequately capture the graph structure for Transformers to succeed.

We use a learned relative positional encoding scheme initialized with random walk probabilities that is related to previously used positional encodings (Dwivedi et al., 2021; Li et al., 2020; Mialon et al., 2021). Going beyond previous work, we give theoretical justification for the use of random walk probabilities — with an appropriate architecture, they are more expressive than shortest path distances, and can capture large classes of graph propagation matrices.

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be the adjacency matrix of a graph $(\mathcal{V}, \mathcal{E})$ with n nodes, and let $\mathbf{D}$ be the diagonal degree matrix. Define $\mathbf{M} := \mathbf{D}^{-1}\mathbf{A}$, and note that $\mathbf{M}_{ij}$ is the probability that i hops to j in one step of a simple random walk. The proposed relative random walk probabilities (RRWP) initial positional encoding is defined for each pair of nodes $i, j \in \mathcal{V}$ as follows:

$$\mathbf{P}_{i,j} = [\mathbf{I}, \mathbf{M}, \mathbf{M}^2, \ldots, \mathbf{M}^{K-1}]_{i,j} \in \mathbb{R}^K, \qquad (1)$$

in which $\mathbf{I}$ is the identity matrix. For any node $i \in \mathcal{V}$, the diagonal $\mathbf{P}_{i,i}$ can additionally be utilized as an initial node-level structural encoding, which is the same as the Random Walk Structural Encodings (RWSE) used in past work (Dwivedi et al., 2021; Rampášek et al., 2022). The parameter $K \in \mathbb{N}$ controls the maximum length of random walks considered.

Importantly, we use RRWP as an initialization of learned relative positional encodings in our architecture. The tensor $\mathbf{P}$ can be updated by an elementwise $\mathrm{MLP} : \mathbb{R}^K \to \mathbb{R}^d$ to get new relative positional encodings $\mathrm{MLP}(\mathbf{P}_{i,j,:})$, and also is updated with other information by the attention layers in our Transformer. This is essential for expressive power, and also may provide a useful inductive bias, as in Proposition 3.1 we show that this updating allows us to recover important propagation matrices.

Visualization of RRWP To further justify and build intuition for initializing with RRWP positional encodings, we

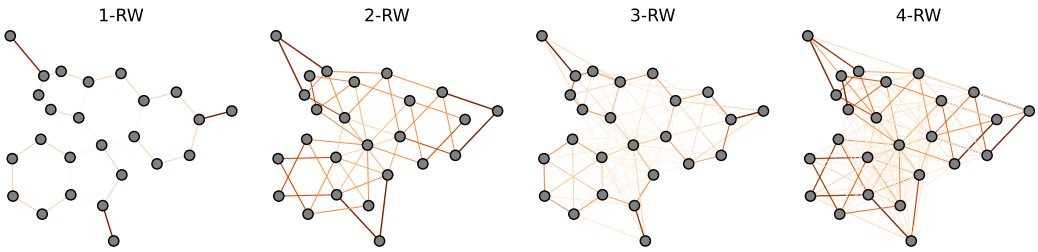


Figure 1. RRWP visualization for the fluorescein molecule, up to the 4th power. Thicker and darker edges indicate higher edge weight. Probabilities for longer random walks reveal higher-order structures (e.g., the cliques evident in 3-RW and the star patterns in 4-RW).

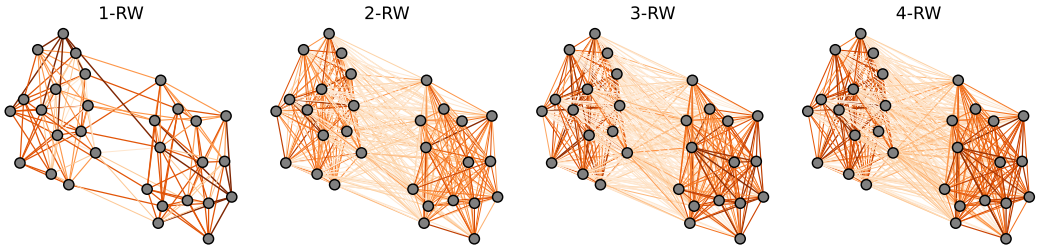


Figure 2. RRWP visualization for a sample from a stochastic block model with 2 communities, up to the 4th power. Probabilities for longer random walks better highlight the community structure and reduce bottlenecks.

demonstrate that it indeed captures useful information on graphs by example. In Figure 1 and Figure 2 we visualize RRWP on two example graphs: the molecular graph for fluorescein and a sample graph from a stochastic block model with 2 communities. For the molecule, we see that RRWP captures higher-order structural information in the longer random walks. In the stochastic block model, longer random walks better reveal the community structure compared to the original graph topology. Moreover, it reduces bottlenecks in the graph, which may be important for certain applications (Alon & Yahav, 2020; Topping et al., 2022).

3.1.1. Theory: RRWP + MLP is Expressive

Given our initial RRWP positional encodings $\mathbf{P}$, we learn new positional encodings end-to-end with an MLP. We show that this combination is provably expressive: this learned positional encoding can approximate shortest path distances or general classes of graph propagation matrices up to an arbitrary $\epsilon > 0$ accuracy. This shows that we generalize methods like Graphormer (Ying et al., 2021), and various message-passing propagations (Gasteiger et al., 2019; Xu et al., 2019).

Proposition 3.1. *For any $n \in \mathbb{N}$, let $\mathbb{G}_n \subseteq \{0,1\}^{n\times n}$ denote all adjacency matrices of n-node graphs. For $K \in \mathbb{N}$, and $\mathbf{A} \in \mathbb{G}_n$ consider the RRWP: $\mathbf{P} = [\mathbf{I}, \mathbf{M}, \ldots, \mathbf{M}^{K-1}] \in \mathbb{R}^{n\times n\times K}$. Then for any $\epsilon > 0$, there exists an $\mathrm{MLP} : \mathbb{R}^K \to \mathbb{R}$ acting independently across each n dimension such that $\mathrm{MLP}(\mathbf{P})$ approximates any of the following to within ϵ error:*

(a) $\mathrm{MLP}(\mathbf{P})_{ij} \approx \mathrm{SPD}_{K-1}(i,j)$

(b) $\mathrm{MLP}(\mathbf{P}) \approx \sum_{k=0}^{K-1} \theta_k (\mathbf{D}^{-1}\mathbf{A})^k$

(c) $\mathrm{MLP}(\mathbf{P}) \approx \theta_0 \mathbf{I} + \theta_1 \mathbf{A}$,

in which $\mathrm{SPD}_{K-1}(i,j)$ is the $K-1$ truncated shortest path distance, and $\theta_k \in \mathbb{R}$ are arbitrary coefficients.

The proof is given in Appendix C.2. (a) shows that, if we use an MLP with K-hop RRWP input, then we can capture all shortest path distances for nodes of up to $K-1$ hops away from each other. In particular, for $K = n$, we recover all shortest path distances (with disconnected nodes getting a distance of n, which is higher than the maximum distance $n-1$ between connected nodes). (b) and (c) show that we can capture many types of graph propagations. As special cases, we can capture sum aggregation [bullet (c) with $\theta_0 = 0, \theta_1 = 1$], mean aggregation [bullet (b) with $K = 2$, $\theta_1 = 1$, other $\theta_i = 0$], K-truncated personalized PageRank (PPR) [bullet (b) with $\theta_k = \alpha(1-\alpha)$ for some $\alpha \in (0,1)$], and K-truncated heat kernels [bullet (b) with $\theta_k = \exp(-\tau)\cdot\tau^k/k!$ for $\tau > 0$]. Such graph propagations may provide useful inductive biases, and a synthetic experiment in Section 4.3 shows that GRIT is highly capable of learning attention mechanisms that match target graph propagations, while other Graph Transformers are less capable.

3.2. Flexible Attention Mechanism with Absolute and Relative Representations

Most current designs of the self-attention mechanism are based on node-token-level PE and node-token-level representations, but this does not fully capture the relative positional information between pairs of nodes. A recent study by Brody et al. (2022) also reveals that some attention mechanisms (e.g., GAT (Veličković et al., 2018) and scaled dot-product attention (Vaswani et al., 2017)) are not sufficiently *flexible* to attend to specific tokens.

Therefore, we propose a new way to compute attention scores by conditioning on (learned) relative representations of node-pairs, which combines the strengths of the general conditioning layer (Perez et al., 2018) and GATv2 (Brody et al., 2022).

In each transformer layer, we update node representations $\mathbf{x}_i, \forall i \in \mathcal{V}$ and node-pair representations $\mathbf{e}_{i,j}, \forall i,j \in \mathcal{V}$. First, we initialize these using the initial node features and our RRWP positional encodings: $\mathbf{x}_i = [\mathbf{x}'_i \| \mathbf{P}_{i,i}] \in \mathbb{R}^{d_h+K}$ and $\mathbf{e}_{i,j} = [\mathbf{e}'_{i,j} \| \mathbf{P}_{i,j}] \in \mathbb{R}^{d_e+K}$, where $\mathbf{x}'_i \in \mathbb{R}^{d_h}$ and $\mathbf{e}'_{i,j} \in \mathbb{R}^{d_e}$ are observed node and edge attributes, which can be dropped if not present in the data. We set $\mathbf{e}'_{i,j} = \mathbf{0}$ if there is no observed edge from i to j in the original graph. The attention computation is defined as follows:

$$
\begin{aligned}
\hat{\mathbf{e}}_{i,j} &= \sigma\Big(\rho\left((\mathbf{W}_{\mathrm{Q}}\mathbf{x}_i + \mathbf{W}_{\mathrm{K}}\mathbf{x}_j) \odot \mathbf{W}_{\mathrm{Ew}}\mathbf{e}_{i,j}\right) \\
&\qquad\qquad + \mathbf{W}_{\mathrm{Eb}}\mathbf{e}_{i,j}\Big) \in \mathbb{R}^{d'}, \\
\alpha_{ij} &= \mathrm{Softmax}_{j\in\mathcal{V}}(\mathbf{W}_{\mathrm{A}}\hat{\mathbf{e}}_{i,j}) \in \mathbb{R}, \\
\hat{\mathbf{x}}_i &= \sum_{j\in\mathcal{V}} \alpha_{ij} \cdot (\mathbf{W}_{\mathrm{V}}\mathbf{x}_j + \mathbf{W}_{\mathrm{Ev}}\hat{\mathbf{e}}_{i,j}) \in \mathbb{R}^{d},
\end{aligned}
\tag{2}
$$

where σ is a non-linear activation (ReLU by default); $\mathbf{W}_{\mathrm{Q}}, \mathbf{W}_{\mathrm{K}}, \mathbf{W}_{\mathrm{Ew}}, \mathbf{W}_{\mathrm{Eb}} \in \mathbb{R}^{d'\times d}$, $\mathbf{W}_{\mathrm{A}} \in \mathbb{R}^{1\times d'}$ and $\mathbf{W}_{\mathrm{V}}, \mathbf{W}_{\mathrm{Ev}} \in \mathbb{R}^{d\times d'}$ are learnable weight matrices; $\odot$ indicates elementwise multiplication; and $\rho(\mathbf{x}) := (\mathrm{ReLU}(\mathbf{x}))^{1/2} - (\mathrm{ReLU}(-\mathbf{x}))^{1/2}$ is the signed-square-root, which stabilizes training by reducing the magnitude of large inputs. We also include biases in our implementation, but they are omitted here for simplicity. Note that we update the pair representations $\mathbf{e}_{i,j}$, so our Transformer is capable of updating the positional encodings. In particular, our Transformer is capable of applying an elementwise MLP to $\mathbf{P}$, as we showed was useful in Proposition 3.1.

Similarly to other self-attention mechanisms, our proposed attention mechanism can be extended to multiple heads (say, N_h heads) by assigning different weight matrices for different heads. We perform the above computations for different heads $h \in \{1, \dots, N_h\}$ to get representations $\hat{\mathbf{x}}_i^h$ and $\hat{\mathbf{e}}_{i,j}^h$, then combine the different heads as follows:

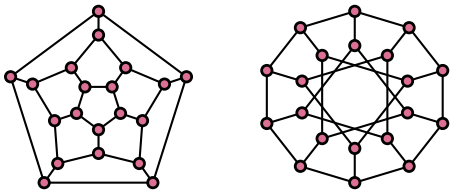


Figure 3. (Left) Dodecahedron graph. (Right) Desargues graph. GD-WL with RRWP can distinguish these two graphs, but GD-WL with SPD cannot.

$$
\begin{aligned}
\mathbf{x}_i^{\mathrm{out}} &= \sum_{h=1}^{N_h} \mathbf{W}_{\mathrm{O}}^h \hat{\mathbf{x}}_i^h \in \mathbb{R}^d, \\
\mathbf{e}_{ij}^{\mathrm{out}} &= \sum_{h=1}^{N_h} \mathbf{W}_{\mathrm{Eo}}^h \hat{\mathbf{e}}_{ij}^h \in \mathbb{R}^d,
\end{aligned}
\tag{3}
$$

where $\mathbf{W}_{\mathrm{O}}^h, \mathbf{W}_{\mathrm{Eo}}^h \in \mathbb{R}^{d\times d'}$ are output weight matrices for each head h.

3.2.1. Theory: RRWP is more expressive than SPD in Transformers

We can use recently proposed Weisfeiler-Leman-like graph isomorphism tests to demonstrate that RRWP within a Transformer architecture is strictly more expressive than the commonly used shortest path distances (SPD). Recently, Zhang et al. (2023) proposed the Generalized Distance Weisfeiler-Leman Test (GD-WL) — a graph isomorphism test based on updating node colors that incorporates distances. Let G be a graph with vertex set $\mathcal{V}$, $d_G(v,u)$ denote a distance between nodes v and u, and $\chi_G^0(v)$ be an initial color of v. Then GD-WL updates node colors as

$$
\chi_G^t(v) = \mathrm{hash}(\{\!\{(d_G(v,u), \chi_G^{t-1}(u)) : u \in \mathcal{V}\}\!\}). \tag{4}
$$

The multiset of final node colors $\{\!\{\chi_G^T(v) : v \in \mathcal{V}\}\!\}$ at iteration T is hashed to get a graph color. Zhang et al. (2023) use GD-WL to analyze a Graph Transformer architecture that uses $d_G(v,u)$ as relative positional encodings. They show that setting $d_G(v,u)$ to the shortest path distance makes it possible to solve edge biconnectivity problems. We can show that if we choose $d_G(v,u)$ to be our relative random walk encoding $d_G(v,u) = \mathbf{P}_{vu} \in \mathbb{R}^n$ (with $K = n$), the GD-WL using this distance is more powerful than the GD-WL using shortest path distances (SPD) [4].

Proposition 3.2. *GD-WL with RRWP distances is strictly stronger than GD-WL with shortest path distances.*

[4] We need to generalize GD-WL to allow vector-valued distances, but this is easily handled by the hash function

The proof is given in Appendix C.1. We first show that GD-WL with RRWP can distinguish any two graphs that GD-WL with SPD can. Then we show that GD-WL with RRWP can distinguish the Dodecahedron and Desargues graphs — plotted in Figure 3 — whereas Zhang et al. (2023) showed that GD-WL with SPD cannot distinguish these.

3.3. Injecting Degree Information

Attention mechanisms are innately invariant to node degrees, analogously to mean-aggregation; this introduces extra ambiguities and hence reduces expressive power in processing graph-structured data (Xu et al., 2019; Corso et al., 2020). Therefore, we introduce an adaptive degree-scaler (Corso et al., 2020) to our attention mechanism to maintain degree information.

After the computation of node representations in (3), we inject degree information into the node representations as follows:

$$\mathbf{x}_i^{\text{out}'} := \mathbf{x}_i^{\text{out}} \odot \boldsymbol{\theta}_1 + \big(\log(1+d_i) \cdot \mathbf{x}_i^{\text{out}} \odot \boldsymbol{\theta}_2\big) \in \mathbb{R}^d\,, \tag{5}$$

where d_i is the degree of node i, and $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d$ are learnable weights. Like other Transformer architectures, we follow this with a standard feed-forward network (FFN) to update the node representations.

To properly include degree information, we apply batch normalization (Ioffe & Szegedy, 2015) instead of the standard layer normalization (Ba et al., 2016; Vaswani et al., 2017) to the outputs of self-attention modules and FFNs. This is because layer normalization applied to each node representation cancels out the effect from degree-scalers or sum-aggregators. We capture this in the following proposition, which we prove in Appendix C.3.

Proposition 3.3. *Sum-aggregated node representations, degree-scaled node representations, and mean-aggregated node representations all have the same value after application of a LayerNorm on node representations.*

4. Experimental Results

4.1. Benchmarking GRIT

We evaluate our proposed method on five benchmarks from the Benchmarking GNNs work (Dwivedi et al., 2022a) and two benchmarks from the recently developed Long-Range Graph Benchmark (Dwivedi et al., 2022b). These benchmarks cover diverse graph learning tasks including node classification, graph classification, and graph regression; they are especially focused on graph structure encoding, node clustering, and learning long-range dependencies. In addition, we also conduct experiments on the larger datasets ZINC-full graphs ($\sim$ 250,000 graphs) (Irwin et al., 2012) and PCQM4Mv2 ($\sim$ 3,700,000 graphs) (Hu et al., 2021). Further details concerning the experimental setup can be found in Appendix B.

Baselines We primarily compare our methods with the recent SOTA hybrid Graph Transformer, GraphGPS (Rampášek et al., 2022), as well as a number of prevalent graph-learning models: popular message-passing neural networks (GCN (Kipf & Welling, 2017), GIN (Xu et al., 2019) and its variant with edge-features (Hu et al., 2020), GAT (Veličković et al., 2018), GatedGCN (Bresson & Laurent, 2018), GatedGCN-LSPE (Dwivedi et al., 2021), PNA (Corso et al., 2020)); Graph Transformers (Graphormer (Ying et al., 2021), K-Subgraph SAT (Chen et al., 2022), EGT (Hussain et al., 2022), SAN (Kreuzer et al., 2021), Graphormer-URPE (Luo et al., 2022), Graphormer-GD (Zhang et al., 2023)); and other recent Graph Neural Networks with SOTA performance (DGN (Beani et al., 2021), GSN (Bouritsas et al., 2022), CIN (Bodnar et al., 2021), CRaW1 (Toenshoff et al., 2021), GIN-AK+ (Zhao et al., 2021b)).

Benchmarks from Benchmarking GNNs (Dwivedi et al., 2022a). We first benchmark our method on five datasets from Benchmarking GNNs (Dwivedi et al., 2022a): ZINC, MNIST, CIFAR10, PATTERN, and CLUSTER, following the experimental setting of GraphGPS (Rampášek et al., 2022) ($\sim$ 500K parameter limit for ZINC, PATTERN, and CLUSTER; $\sim$ 100K parameter limit for MNIST and CIFAR10). Results are shown in Table 1; we report the mean and standard deviation across 4 runs with different random seeds.

We show that our model has the best mean performance for four of the five datasets with statistically significant improvement. In the remaining dataset, our model reaches the second-best performance without a statistically significant difference compared to the best performer. These results showcase the ability of GRIT to outperform a variety of methods on small to medium-sized datasets, including MPNNs, expressive higher-order GNNs, and Graph Transformers.

Long-Range Graph Benchmark (Dwivedi et al., 2022b). Next, we evaluate our method on the recently proposed Long-Range Graph Benchmark (LRGB). We conduct experiments on the two peptide graph benchmarks from LRGB, namely Peptides-func and Peptides-struct, which are 10-task multilabel classification and 11-task regression tasks, respectively. Results are shown in Table 2. On both datasets, our method obtains the best mean performance, outperforming MPNNs and Graph Transformers — this demonstrates that our model is capable of learning long range interactions.

Table 1. Test performance in five benchmarks from (Dwivedi et al., 2022a). Shown is the mean ± s.d. of 4 runs with different random seeds. Highlighted are the top first, second, and third results. # Param $\sim 500K$ for ZINC, PATTERN, CLUSTER and $\sim 100K$ for MNIST and CIFAR10. * indicates statistically significant difference against the second-best result from the two-sample one-tailed t-test.

Model	ZINC	MNIST	CIFAR10	PATTERN	CLUSTER
	MAE↓	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑
GCN	0.367 ± 0.011	90.705 ± 0.218	55.710 ± 0.381	71.892 ± 0.334	68.498 ± 0.976
GIN	0.526 ± 0.051	96.485 ± 0.252	55.255 ± 1.527	85.387 ± 0.136	64.716 ± 1.553
GAT	0.384 ± 0.007	95.535 ± 0.205	64.223 ± 0.455	78.271 ± 0.186	70.587 ± 0.447
GatedGCN	0.282 ± 0.015	97.340 ± 0.143	67.312 ± 0.311	85.568 ± 0.088	73.840 ± 0.326
GatedGCN-LSPE	0.090 ± 0.001	–	–	–	–
PNA	0.188 ± 0.004	97.94 ± 0.12	70.35 ± 0.63	–	–
DGN	0.168 ± 0.003	–	72.838 ± 0.417	86.680 ± 0.034	–
GSN	0.101 ± 0.010	–	–	–	–
CIN	0.079 ± 0.006	–	–	–	–
CRaW1	0.085 ± 0.004	97.944 ± 0.050	69.013 ± 0.259	–	–
GIN-AK+	0.080 ± 0.001	–	72.19 ± 0.13	86.850 ± 0.057	–
SAN	0.139 ± 0.006	–	–	86.581 ± 0.037	76.691 ± 0.65
Graphormer	0.122 ± 0.006	–	–	–	–
K-Subgraph SAT	0.094 ± 0.008	–	–	86.848 ± 0.037	77.856 ± 0.104
EGT	0.108 ± 0.009	98.173 ± 0.087	68.702 ± 0.409	86.821 ± 0.020	79.232 ± 0.348
Graphormer-URPE	0.086 ± 0.007	–	–	–	–
Graphormer-GD	0.081 ± 0.009	–	–	–	–
GPS	0.070 ± 0.004	98.051 ± 0.126	72.298 ± 0.356	86.685 ± 0.059	78.016 ± 0.180
GRIT (ours)	$0.059 \pm 0.002^*$	98.108 ± 0.111	$76.468 \pm 0.881^*$	$87.196 \pm 0.076^*$	$80.026 \pm 0.277^*$

Table 2. Test performance on two benchmarks from long-range graph benchmarks (LRGB) (Dwivedi et al., 2022b). Shown is the mean ± s.d. of 4 runs with different random seeds. Highlighted are the top first, second, and third results. # Param $\sim 500K$ for both datasets. * indicates statistical significance against the second-best results from the Two-sample One-tailed T-Test.

Model	Peptides-func	Peptides-struct
	AP↑	MAE↓
GCN	0.5930 ± 0.0023	0.3496 ± 0.0013
GINE	0.5498 ± 0.0079	0.3547 ± 0.0045
GatedGCN	0.5864 ± 0.0035	0.3420 ± 0.0013
GatedGCN+RWSE	0.6069 ± 0.0035	0.3357 ± 0.0006
Transformer+LapPE	0.6326 ± 0.0126	0.2529 ± 0.0016
SAN+LapPE	0.6384 ± 0.0121	0.2683 ± 0.0043
SAN+RWSE	0.6439 ± 0.0075	0.2545 ± 0.0012
GPS	0.6535 ± 0.0041	0.2500 ± 0.0012
GRIT (ours)	$0.6988 \pm 0.0082^*$	$0.2460 \pm 0.0012^*$

Table 3. Test performance on ZINC-full (Irwin et al., 2012). # Param $\sim 500K$. Shown is the mean ± s.d. of 4 runs with different random seeds. Highlighted are the top first, second, and third results.

Method	Model	ZINC-full (MAE ↓)
MPNNs	GIN	0.088 ± 0.002
	GraphSAGE	0.126 ± 0.003
	GAT	0.111 ± 0.002
	GCN	0.113 ± 0.002
	MoNet	0.090 ± 0.002
Higher-order GNNs	δ-2-GNN	0.042 ± 0.003
	δ-2-LGNN	0.045 ± 0.006
PE-GNN	SignNet	0.024 ± 0.003
Graph Transformers	Graphormer	0.052 ± 0.005
	Graphormer-URPE	0.028 ± 0.002
	Graphormer-GD	0.025 ± 0.004
	GRIT (ours)	0.023 ± 0.001

ZINC-full Dataset We also test our model on the ZINC-full dataset (Irwin et al., 2012), which is the full version of ZINC that has 250,000 graphs. Besides the MPNNs and Graph Transformers, we also compare our method with other domain agnostic methods like higher-order GNNs (δ-2-GNN, δ-2-LGNN (Morris et al., 2020)) as well as PE-enhanced GNNs (SignNet (Lim et al., 2023)). We see that GRIT outperforms various classes of methods, and achieves the best mean performance of all methods.

PCQM4Mv2 Large-scale Graph Regression Benchmark (Hu et al., 2021) Further, we conduct an experiment on the PCQM4Mv2 large-scale graph regression benchmark of 3.7M graphs (Hu et al., 2021), which is currently one of the largest molecular datasets (Table. 4). We compare our method against MPNNs (GCN (Kipf & Welling, 2017), GIN (Xu et al., 2019) with/without virtual nodes) as well as several Graph Transformers (GRPE (Park et al., 2022), Graphormer (Ying et al., 2021), TokenGT (Kim et al., 2022)

and GraphGPS (Rampášek et al., 2022)). Following the protocol of Rampášek et al. (2022), we treated the validation set of the dataset as a test set, since the *Test-dev* set labels are private. The result of a single random seed run is reported due to the size of the dataset, following previous works (Rampášek et al., 2022; Kim et al., 2022). Our model can reach a comparable performance to GraphGPS (best) and Graphormer (third best) while using fewer learnable parameters. We did not conduct any hyperparameter search due to the limit of time, and instead adopted the values from GraphGPS (Rampášek et al., 2022).

Table 4. Test performance on PCQM4Mv2 (Hu et al., 2021) dataset. Shown is the result of a single run due to the computation constraint. Highlighted are the top first, second, and third results.

Method	Model	Valid. (MAE ↓)	# Param
MPNNs	GCN	0.1379	2.0M
	GCN-virtual	0.1153	4.9M
	GIN	0.1195	3.8M
	GIN-virtual	0.1083	6.7M
Graph Transformers	GRPE	0.0890	46.2M
	Graphormer	**0.0864**	48.3M
	TokenGT (ORF)	0.0962	48.6M
	TokenGT (Lap)	0.0910	48.5M
	GPS-small	0.0938	6.2M
	GPS-medium	**0.0858**	19.4M
	GRIT (ours)	**0.0859**	16.6M

Table 5. Ablations on design choices of our architecture on ZINC (Dwivedi et al., 2022a). Substituting other design choices decreases the performance of our model. Shown is the mean ± s.d. of 4 runs with different random seeds. A → B stands for using B instead of A.

ZINC	MAE ↓
GRIT (ours)	$\mathbf{0.059 \pm 0.002}$
- Remove degree scaler	0.076 ± 0.002
- Remove the update of RRWP	0.066 ± 0.005
- Global-attn. → Sparse-attn.	0.066 ± 0.002
- Degree scaler → Degree encoding	0.072 ± 0.005
- GRIT-attn. → Graphormer-attn.	0.117 ± 0.028
- RRWP → RWSE	0.081 ± 0.010
- RRWP → SPDPE	0.067 ± 0.002

4.2. Ablations

To determine the utility of our architectural design choices, we conduct several ablation experiments on ZINC. Table 5 shows the results. Removing degree scalers, removing the update mechanism of RRWP, substituting global attention with sparse attention, replacing the degree scalers with degree encoding from Graphormer (Ying et al., 2021), replacing our attention mechanism with the attention mechanism used in Graphormer (no PE update), and substituting RRWP with RWSE (Dwivedi et al., 2021) or SPDPE (Ying et al., 2021), all lead to worse performance — this lends credence to our architectural choices.

We also conduct a sensitivity analysis on the parameter K of RRWP on ZINC. The results are shown in Table 6. Notably, our method is SOTA or near SOTA for many choices of K, except for very unreasonable choices like $K = 2$. Note that we keep every other hyperparameter fixed besides K in this experiment, which was originally chosen for $K = 21$. This might explain why other K perform slightly worse.

Table 6. Sensitivity Analysis of K-order RRWP on ZINC (Dwivedi et al., 2022a). Shown is the mean ± s.d. of 4 runs with different random seeds.

K	2	7	14	18	21	24	42
MAE ↓	0.147	0.063	0.063	0.060	0.059	0.060	0.061
±	0.006	0.003	0.004	0.002	0.002	0.002	0.001

Table 7. Synthetic Experiment on Learning to Attend K-hop Neighborhoods

MAE ≤ 1 ↓	1-hop	2-hop	3-hop
MeanPool (baseline)	$.083 \pm .015$	$.080 \pm .014$	$.069 \pm .011$
Transf.+RWSE	$.083 \pm .015$	$.080 \pm .014$	$.069 \pm .011$
SAN+LapPE	$.044 \pm .011$	$.042 \pm .010$	$.029 \pm .008$
Graphormer+SPDPE	$.043 \pm .010$	$.034 \pm .010$	$.025 \pm .005$
GRIT (Ours)	$\mathbf{.001 \pm .001}$	$\mathbf{.001 \pm .001}$	$\mathbf{.007 \pm .004}$

4.3. Synthetic Experiment: Can Our Attention Module Learn to Attend to K-hop Neighbors?

We conduct a synthetic experiment to study the ability of our proposed attention modules to emulate a general class of graph propagation matrices, in comparison to existing Graph Transformers. Concretely, we consider single-layer, single-head attention modules with the corresponding positional encoding as the only input. We experiment with GRIT and a set of popular Graph Transformers: vanilla Transformer+RWSE (the Transformer branch of GraphGPS (Rampášek et al., 2022)), SAN+LapPE* (global attention component with the LapPE transformer) (Kreuzer et al., 2021), and Graphormer+SPDPE (Ying et al., 2021).

We train these attention mechanisms to output a row-normalized adjacency matrix for $(k = 1, 2, 3)$-hop neighborhoods, using the l_1-loss as the training objective. More precisely, the targets are generated as follows. We choose a $k \in \{1, 2, 3\}$, compute $\mathbf{A}^k$ for an adjacency matrix $\mathbf{A}$, round nonzero values to have the value 1, and then normalize each row to sum to one.

With 20 graphs randomly sampled from the ZINC dataset,

we train each method for 2000 epochs on each graph separately and compute the (training) mean absolute errors across graphs. From the results (Table 7), we observe that SAN+LapPE* and Graphormer+SPDPE perform similarly in learning to attend to a specific type of neighborhood, while Transformer+RWSE performs much worse; this may help explain why MPNNs are so essential for GraphGPS, according to its ablation studies (Table B.1 (Rampášek et al., 2022)). GRIT significantly outperforms other baselines by an order of magnitude, showing that our method can well approximate a general class of graph propagations, as theoretically expected (due to Proposition 3.1).

In addition, we also visualize the attention scores for a single graph of the synthetic experiments (see Figure 5 in Appendix B.5), which agrees with the quantitative results. Whereas other attention mechanisms qualitatively struggle at matching the target sparsity pattern or attention magnitudes, our GRIT attention mechanism succeeds at learning to match both sparsity and magnitudes.

5. Conclusion

Observing the performance gap of Graph Transformers between small and large-scale datasets, we argue for the importance of graph inductive biases in Graph Transformers. Motivated by promoting inductive biases in Graph Transformers without message-passing, we propose GRIT, based on three theoretically and empirically motivated design choices for incorporating graph inductive biases. Our learned relative positional encodings initialized with RRWP along with our flexible attention mechanism allow for an expressive model that can provably capture shortest path distances and general families of graph propagations. Theoretical results show that the RRWP initialization is strictly more expressive than shortest path distances when used in the GD-WL graph isomorphism test (Zhang et al., 2023), and a synthetic experiment shows that our flexible attention mechanism is indeed able to learn graph propagation matrices that other Graph Transformers are not as capable of learning. Our GRIT model achieves state-of-the-art performance across a wide range of graph datasets, showing that data complexity can be improved for Graph Transformers without integrating local message-passing modules. Nonetheless, GRIT is not the final chapter for graph inductive biases in Transformers; future work can address limitations of our work, such as GRIT's n^2 scaling for updating pair representations, and lack of upper bounds on expressive power.

6. Acknowledgments

DL is supported by an NSF Graduate Fellowship. CL is supported by Meta AI [5]. We would also like to thank the Royal Academy of Engineering and FiveAI.

References

Alon, U. and Yahav, E. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *Proc. Int. Conf. Learn. Representations*, September 2020.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer Normalization. In *NIPS 2016 Deep Learning Symposium*, August 2016.

Beani, D., Passaro, S., Létourneau, V., Hamilton, W., Corso, G., and Lió, P. Directional Graph Networks. In *Proc. Int. Conf. Mach. Learn.*, pp. 748–758. PMLR, July 2021.

Bevilacqua, B., Frasca, F., Lim, D., Srinivasan, B., Cai, C., Balamurugan, G., Bronstein, M. M., and Maron, H. Equivariant Subgraph Aggregation Networks. In *Proc. Int. Conf. Learn. Representations*, March 2022.

Bodnar, C., Frasca, F., Wang, Y., Otter, N., Montufar, G. F., Lió, P., and Bronstein, M. Weisfeiler and Lehman Go Topological: Message Passing Simplicial Networks. In *Proc. Int. Conf. Mach. Learn.*, pp. 1026–1037. PMLR, July 2021.

Bouritsas, G., Frasca, F., Zafeiriou, S. P., and Bronstein, M. Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting. *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–1, 2022. ISSN 1939-3539. doi: 10.1109/TPAMI.2022.3154319.

Bresson, X. and Laurent, T. Residual Gated Graph ConvNets. *arXiv*, April 2018.

Brody, S., Alon, U., and Yahav, E. How Attentive are Graph Attention Networks? In *Proc. Int. Conf. Learn. Representations)*, 2022.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. May 2021.

Chen, D., O'Bray, L., and Borgwardt, K. Structure-Aware Transformer for Graph Representation Learning. In *Proc. Int. Conf. Mach. Learn.*, pp. 3469–3489, June 2022.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Veličković, P. Principal Neighbourhood Aggregation for Graph Nets. In *Adv. Neural Inf. Process. Syst.*, December 2020.

[5] Meta has no relationships whatsoever with the other funding sponsors.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proc. Annu. Meeting Assoc. Comput. Linguist.*, pp. 2978–2988, July 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. Annu. Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. (NAACL-HILT)*, May 2019.

Diao, C. and Loynd, R. Relational attention: Generalizing transformers for graph-structured tasks. *arXiv preprint arXiv:2210.05062*, 2022.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proc. Int. Conf. Learn. Representations*, September 2020.

Dwivedi, V. P. and Bresson, X. A Generalization of Transformer Networks to Graphs. In *Proc. AAAI Workshop Deep Learn. Graphs: Methods Appl.*, January 2021.

Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Graph Neural Networks with Learnable Structural and Positional Representations. In *Proc. Int. Conf. Learn. Representations*, September 2021.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking Graph Neural Networks. *J. Mach. Learn. Res.*, December 2022a.

Dwivedi, V. P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A. T., and Beaini, D. Long Range Graph Benchmark. In *Adv. Neural Inf. Process. Syst.*, December 2022b.

Feldman, O., Boyarski, A., Feldman, S., Kogan, D., Mendelson, A., and Baskin, C. Weisfeiler and leman go infinite: Spectral and combinatorial pre-colorings. In *Proc. Int. Conf. Learn. Representations Workshop Geom. topol. Representation Learn.*, 2022.

Gasteiger, J., Weiß enberger, S., and Günnemann, S. Diffusion Improves Graph Learning. In *Adv. Neural Inf. Process. Syst.*, volume 32. Curran Associates, Inc., 2019.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural Message Passing for Quantum Chemistry. In *Proc. Int. Conf. Mach. Learn.*, June 2017.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, 1989.

Hu, W., Liu*, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. Strategies for Pre-training Graph Neural Networks. In *Proc. Int. Conf. Learn. Representations*, March 2020.

Hu, W., Fey, M., Ren, H., Nakata, M., Dong, Y., and Leskovec, J. Ogb-lsc: A large-scale challenge for machine learning on graphs. In *Adv. Neural Inf. Process. Syst. Datasets Benchmarks Track*, 2021.

Hussain, M. S., Zaki, M. J., and Subramanian, D. Global Self-Attention as a Replacement for Graph Convolution. In *Proc. ACM SIGKDD Int. Conf. Knowl Discov. Data Min. (KDD)*, August 2022. doi: 10.1145/3534678.3539296.

Ioffe, S. and Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proc. Int. Conf. Mach. Learn.*, pp. 448–456. PMLR, June 2015.

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., and Coleman, R. G. ZINC: A Free Tool to Discover Chemistry for Biology. *J. Chem. Inf. Model.*, 52(7):1757–1768, July 2012. ISSN 1549-9596. doi: 10.1021/ci3001277.

Kim, J., Nguyen, D. T., Min, S., Cho, S., Lee, M., Lee, H., and Hong, S. Pure Transformers are Powerful Graph Learners. In *Adv. Neural Inf. Process Syst.*, October 2022.

Kipf, T. N. and Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In *Proc. Int. Conf. Learn. Representations*, 2017.

Kreuzer, D., Beaini, D., Hamilton, W. L., Létourneau, V., and Tossou, P. Rethinking Graph Transformers with Spectral Attention. In *Adv. Neural Inf. Process. Syst.*, May 2021.

Li, P., Wang, Y., Wang, H., and Leskovec, J. Distance Encoding: Design Provably More Powerful Neural Networks for Graph Representation Learning. In *Adv. Neural Inf. Process. Syst.*, 2020.

Li, Q., Han, Z., and Wu, X.-M. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In *Proc. AAAI Conf. Artif. Intell.*, pp. 9, 2018.

Lim, D., Robinson, J. D., Zhao, L., Smidt, T., Sra, S., Maron, H., and Jegelka, S. Sign and Basis Invariant Networks for Spectral Graph Representation Learning. In *Proc. Int. Conf. Learn. Representations*, 2023.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows. In *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, pp. 10012–10022, 2021.

Loukas, A. What graph neural networks cannot learn: Depth vs width. In *Proc. Int. Conf. Learn. Representations*, March 2020.

Luo, S., Li, S., Zheng, S., Liu, T.-Y., Wang, L., and He, D. Your transformer may not be as powerful as you expect. In *Adv. Neural Inf. Process. Syst.*, 2022.

Ma, L., Rabbany, R., and Romero-Soriano, A. Graph attention networks with positional embeddings. In *Pacific-Asia Conf. Knowl. Discov. and Data Min.*, pp. 514–527. Springer, 2021.

Masters, D., Dean, J., Klaser, K., Li, Z., Maddrell-Mander, S., Sanders, A., Helal, H., Beker, D., Rampášek, L., and Beaini, D. GPS++: An Optimised Hybrid MPNN/Transformer for Molecular Property Prediction, December 2022.

Mialon, G., Chen, D., Selosse, M., and Mairal, J. GraphiT: Encoding Graph Structure in Transformers, June 2021.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *Proc. AAAI Conf. Artif. Intell.*, volume 33, pp. 4602–4609, July 2019. doi: 10.1609/aaai.v33i01.33014602.

Morris, C., Rattan, G., and Mutzel, P. Weisfeiler and leman go sparse: towards scalable higher-order graph embeddings. In *Adv. Neural Inf. Process. Syst.*, pp. 21824–21840, 2020.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. In *Proc. Int. Conf. Learn. Representations*, 2020.

Park, N. and Kim, S. How Do Vision Transformers Work? In *Proc. Int. Conf. Learn. Representations*, September 2021.

Park, W., Chang, W., Lee, D., Kim, J., and Hwang, S.-w. GRPE: Relative Positional Encoding for Graph Transformer. In *Proc. Int. Conf. Learn. Representations Workshop Mach. Learn. Drug Discov.*, 2022.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. Film: Visual reasoning with a general conditioning layer. In *Proc. AAAI Conf. Artif. Intell.*, volume 32, 2018.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a General, Powerful, Scalable Graph Transformer. In *Adv. Neural Inf. Process. Syst.*, May 2022.

Srinivasan, B. and Ribeiro, B. On the Equivalence between Positional Node Embeddings and Structural Graph Representations. In *Proc. Int. Conf. Learn. Representations*, 2020.

Toenshoff, J., Ritzert, M., Wolf, H., and Grohe, M. Graph learning with 1d convolutions on random walks. *arXiv preprint arXiv:2102.08786*, 2021.

Topping, J., Giovanni, F. D., Chamberlain, B. P., Dong, X., and Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature. In *Proc. Int. Conf. Learn. Representations*, March 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Aidan N Gomez, Kaiser, L., and Polosukhin, I. Attention is All you Need. In *Adv. Neural Inf. Process. Syst.*, volume 30. Curran Associates, Inc., 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph Attention Networks. In *Proc. Int. Conf. Learn. Representations*, February 2018.

Wang, H., Yin, H., Zhang, M., and Li, P. Equivariant and Stable Positional Encoding for More Powerful Graph Neural Networks. In *Proc. Int. Conf. Learn. Representations*, May 2022.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How Powerful are Graph Neural Networks? In *Proc. Int. Conf. Learn. Representations*, February 2019.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. Do Transformers Really Perform Badly for Graph Representation? In *Adv. Neural Inf. Process. Syst.*, 2021.

You, J., Ying, R., and Leskovec, J. Position-aware Graph Neural Networks. In *Proc. Int. Conf. Mach. Learn.*, pp. 7134–7143. PMLR, May 2019.

You, J., Gomes-Selman, J., Ying, R., and Leskovec, J. Identity-aware Graph Neural Networks. In *Proc. AAAI Conf. Artif. Intell.*, February 2021.

Zhang, B., Luo, S., Wang, L., and He, D. Rethinking the expressive power of GNNs via graph biconnectivity. In *Proc. Int. Conf. Learn. Representations*, 2023. URL `https://openreview.net/forum?id=r9hNv76KoT3`.

Zhang, Z., Cui, P., Pei, J., Wang, X., and Zhu, W. Eigen-GNN: A Graph Structure Preserving Plug-in for GNNs. *IEEE Trans. Knowl. Data Eng.*, pp. 1–1, 2021. ISSN 1558-2191. doi: 10.1109/TKDE.2021.3112746.

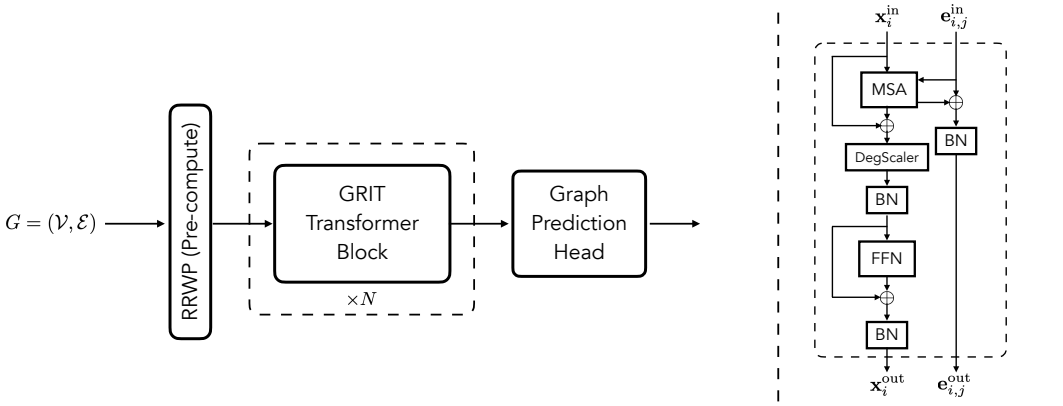
Zhao, J., Dong, Y., Ding, M., Kharlamov, E., and Tang, J. Adaptive Diffusion in Graph Neural Networks. In *Adv. Neural Inf. Process. Syst.*, volume 34, pp. 23321–23333. Curran Associates, Inc., 2021a.

Zhao, L., Jin, W., Akoglu, L., and Shah, N. From stars to subgraphs: Uplifting any gnn with local structure awareness. In *Proc. Int. Conf. Learn. Representations*, 2021b.

A. Model Architecture

A.1. Visualization of the Transformer Architecture of GRIT

In order to put all the conceptual building blocks into one clear visualization, here we provide an overview of the GRIT transformer in Figure 4.



(a) The Overall Architecture of GRIT Transformers

(b) The Architecture of a GRIT Transformer Block

Figure 4. The Architecture of GRIT Transformers. (a) visualize the conceptural relationship between our proposed RRWP feature and the GRIT transformer. (b) shows the detailed design of GRIT transformer block.

B. Experimental Details

B.1. Description of Datasets

A summary of the statistics and characteristics of datasets is shown in Table. 8. The first five datasets are from Dwivedi et al. (2022a), the middle two are from Dwivedi et al. (2022b) and the last is from Hu et al. (2021). Readers are referred to Rampášek et al. (2022) for more details about the datasets.

Table 8. Overview of the graph learning datasets involved in this work (Dwivedi et al., 2022a;b; Irwin et al., 2012; Hu et al., 2021) .

Dataset	**# Graphs**	**Avg. # nodes**	**Avg. # edges**	**Directed**	**Prediction level**	**Prediction task**	**Metric**
ZINC(-full)	12,000 (250,000)	23.2	24.9	No	graph	regression	Mean Abs. Error
MNIST	70,000	70.6	564.5	Yes	graph	10-class classif.	Accuracy
CIFAR10	60,000	117.6	941.1	Yes	graph	10-class classif.	Accuracy
PATTERN	14,000	118.9	3,039.3	No	inductive node	binary classif.	Weighted Accuracy
CLUSTER	12,000	117.2	2,150.9	No	inductive node	6-class classif.	Accuracy
Peptides-func	15,535	150.9	307.3	No	graph	10-task classif.	Avg. Precision
Peptides-struct	15,535	150.9	307.3	No	graph	11-task regression	Mean Abs. Error
PCQM4Mv2	3,746,620	14.1	14.6	No	graph	regression	Mean Abs. Error

B.2. Dataset splits and random seed

Our experiments are conducted on the standard train/validation/test splits of the evaluated benchmarks. For each dataset, we execute 4 runs with different random seeds (0,1,2,3) and report the mean performance and standard deviation.

B.3. Hyperparameters

Due to the limited time and computational resources, we did not perform an exhaustive search or a grid search on the hyperparameters. We mainly follow the hyperparameter setting of GraphGPS (Rampášek et al., 2022) and make slight changes if the number of parameters does not fit in the commonly used parameter budgets. For the benchmarks from Dwivedi

et al. (2022a;b), we follow the most commonly used parameter budgets: up to 500k parameters for ZINC, PATTERN, CLUSTER, Peptides-func and Peptides-struct; and 100k parameters for MNIST and CIFAR10.

The final hyperparameters are presented in Tables. 9 and Tables. 10.

Table 9. Hyperparameters for five datasets from BenchmarkingGNNs (Dwivedi et al., 2022a) and ZINC-full (Irwin et al., 2012)

Hyperparameter	ZINC/ZINC-full	MNIST	CIFAR10	PATTERN	CLUSTER
# Transformer Layers	10	3	3	10	16
Hidden dim	64	52	52	64	48
# Heads	8	4	4	8	8
Dropout	0	0	0	0	0.01
Attention dropout	0.2	0.5	0.5	0.2	0.5
Graph pooling	sum	mean	mean	–	–
PE dim (RW-steps)	21	18	18	21	32
PE encoder	linear	linear	linear	linear	linear
Batch size	32/256	16	16	32	16
Learning Rate	0.001	0.001	0.001	0.0005	0.0005
# Epochs	2000	200	200	100	100
# Warmup epochs	50	5	5	5	5
Weight decay	$1e-5$	$1e-5$	$1e-5$	$1e-5$	$1e-5$
# Parameters	473,473	102,138	99486	477,953	432,206

Table 10. Hyperparameters for two datasets from the Long-range Graph Benchmark (Dwivedi et al., 2022b) and PCQM4Mv2 (Hu et al., 2021)

Hyperparameter	Peptides-func	Peptides-struct	PCQM4Mv2
# Transformer Layers	4	4	16
Hidden dim	96	96	256
# Heads	4	8	8
Dropout	0	0	0.1
Attention dropout	0.5	0.5	0.1
Graph pooling	mean	mean	mean
PE dim (walk-step)	17	24	16
PE encoder	linear	linear	linear
Batch size	32	32	256
Learning Rate	0.0003	0.0003	0.0002
# Epochs	200	200	150
# Warmup epochs	5	5	10
Weight decay	0	0	0
# Parameters	443,338	438,827	15.3M

B.4. Significance Test

We conduct a two-sample one-tailed T-test to compare the results of our method with the second-best model on each dataset. The baselines' results are taken from (Rampášek et al., 2022), with 10 runs for datasets from (Dwivedi et al., 2022a) and 4 runs for datasets from (Dwivedi et al., 2022b).

The statistical tests are conducted using the tools available at `https://www.statskingdom.com/140MeanT2eq.html`.

B.5. Visualization for the Synthetic Experiment

In Figure 5, we visualize the learned attention scores for a single graph from our synthetic experiments in Section 4.3. Recall that the goal of the synthetic experiments was to test the ability of different attention mechanisms and positional encoding schemes to learn to attend to k-hop neighborhoods, using only a single layer of attention and only the PE as input.

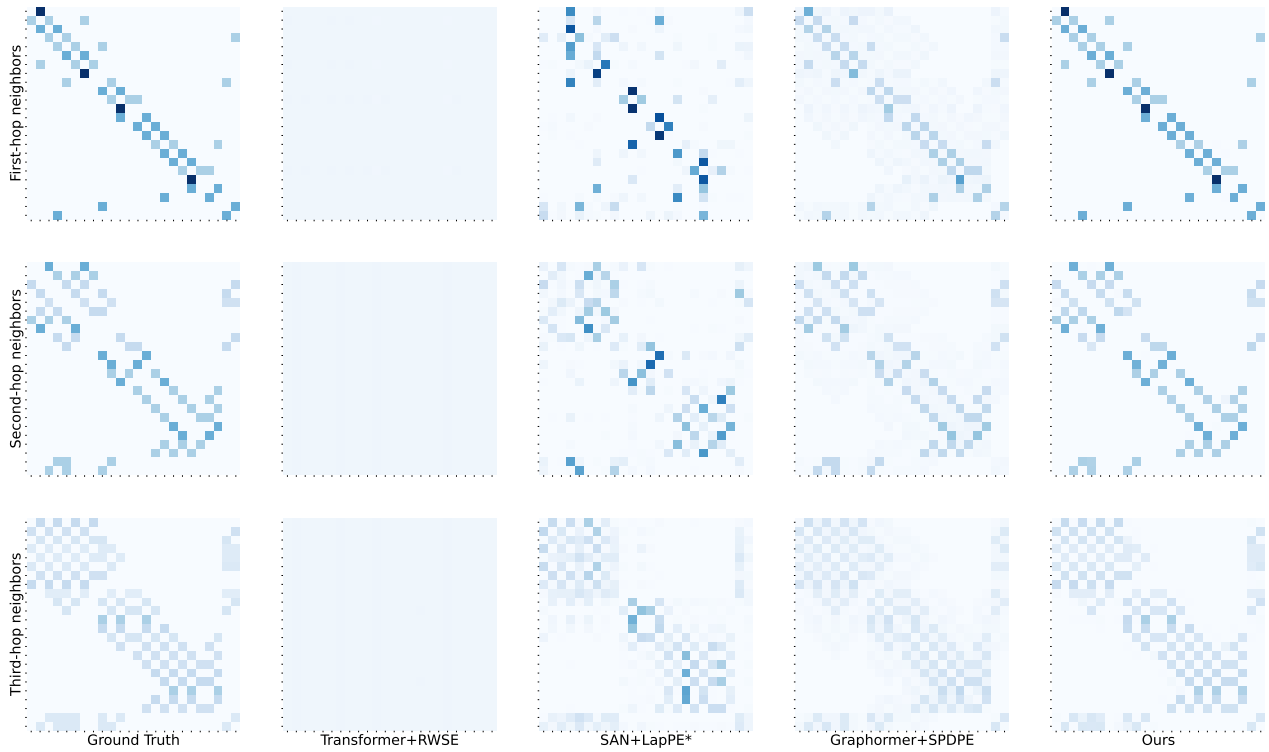


Figure 5. Visualization of learned attention scores for the synthetic experiment on learning to attend to ($k = 1, 2, 3$)-hop neighbors. Our GRIT attention mechanism (far right) is the most successful at matching both the sparsity pattern and attention magnitudes of the target (far left).

Transformer+RWSE struggles to learn to attend to k-hop neighbors. Graphormer+SPDPE can successfully attend to most nodes in k-hop neighborhoods; however, many other nodes are still assigned small attention scores, so the fraction of attention focused on the neighborhoods is diminished. SAN+LapPE* is better at capturing the scales of the target attention scores, but it assigns high attention scores to multiple nodes outside the targeted neighborhoods. In contrast, our GRIT attention mechanism can successfully attend to the nodes in k-hop neighborhoods. We also provide the result table of the synthetic experiment with R^2 metric, shown in Table. 11.

Table 11. Synthetic Experiment on Learning to Attend K-hop Neighborhoods

$R^2 \leq 1$	1-hop	2-hop	3-hop
MeanPool (baseline)	0 ± 0	0 ± 0	0 ± 0
Transf.+RWSE	0 ± 0	0 ± 0	0 ± 0
SAN+LapPE	0.32 ± 0.20	0.31 ± 0.24	0.52 ± 0.21
EGT+SVDPE	0.55 ± 0.21	0.28 ± 0.25	0.26 ± 0.28
Graphormer+SPDPE	0.67 ± 0.09	0.77 ± 0.08	0.80 ± 0.06
GRIT (Ours)	$\mathbf{0.999 \pm 0.001}$	$\mathbf{0.998 \pm 0.004}$	$\mathbf{0.961 \pm 0.035}$

B.6. Asymptotic Complexity, Runtime and GPU Memory

The asymptotic complexities of RRWP and GRIT's attention mechanism are $O(K|\mathcal{V}||\mathcal{E}|)$ and $O(|\mathcal{V}|^2)$ respectively, where K is the number of hops of RRWP, $|\mathcal{E}|$ is the number of edges and $|\mathcal{V}|$ is the number of nodes, matching the asymptotic complexity of most Graph Transformers (Kreuzer et al., 2021; Rampášek et al., 2022; Hussain et al., 2022; Ying et al., 2021). .

We provide the runtime and GPU memory consumption of GRIT and other baselines on ZINC as a reference (Table. 12). The runtime is given by the pipeline of GraphGPS (Rampášek et al., 2022), and the memory usage is counted by the NVIDIA System Management Interface (nvidia-smi).

Table 12. Runtime and GPU memory for SAN (Kreuzer et al., 2021), GraphGPS (Rampášek et al., 2022) and GRIT (Ours) on ZINC with batch size 32. The timing is conducted on a single NVIDIA V100 GPU and 20 threads of Intel(R) Xeon(R) GOld 6140 CPU @ 2.30GH.

ZINC	SAN	GraphGPS	GRIT (Ours)
MAE ↓	0.139 ± 0.006	0.070 ± 0.004	0.059 ± 0.002
PE Precompute-time	10 sec	11 sec	11 sec
GPU Memory	2291 MB	1101 MB	1865 MB
Training time	57.9 sec/epoch	24.3 sec/epoch	29.4 sec/epoch

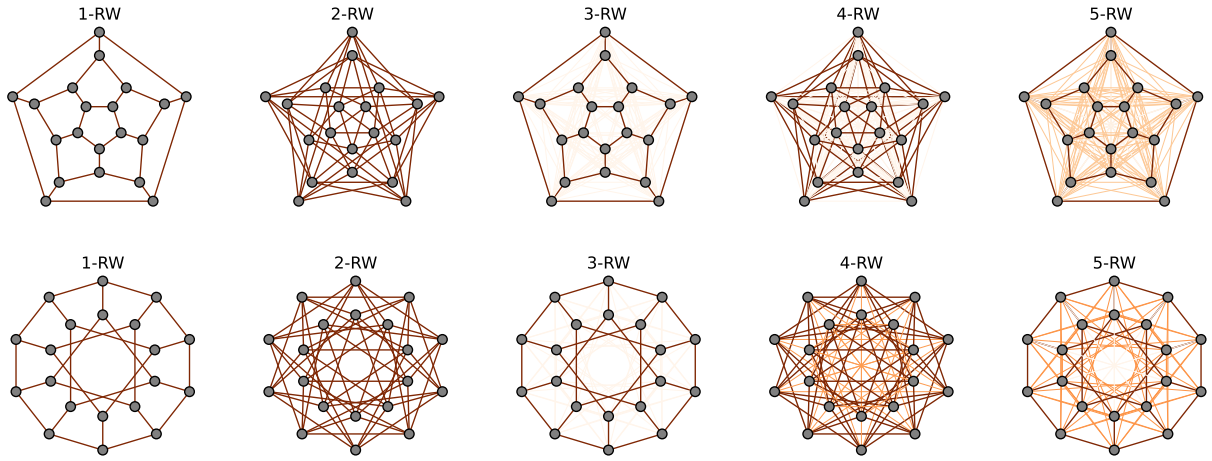


Figure 6. (Top row) RRWP for the Dodecahedron graph. (Bottom row) RRWP for the Desargues graph. This pair of non-isomorphic graphs cannot be distinguished by GD-WL with shortest path distances, but can be distinguished by GD-WL with our RRWP positional encoding.

C. Proofs of Theoretical Results

C.1. GD-WL

Proof of Proposition 3.2. First, we show that GD-WL with RRWP is at least as strong as GD-WL with shortest path distances. Then we give an example of two graphs that GD-WL with shortest path distances cannot distinguish, yet GD-WL with RRWP can.

Let $d_G^{\mathrm{RW}}(v,u) \in \mathbb{R}^n$ be the relative random walk encoding with $K = n$, and $d_G^{\mathrm{SPD}} \in \mathbb{R}$ be the shortest path distance encoding. Then note that

$$\min\{i : \ d_G^{\mathrm{RW}}(v,u)_i \neq 0\} = d_G^{\mathrm{SPD}}(v,u)\}, \tag{6}$$

where the $\min$ takes the value ∞ if $d_G^{\mathrm{RW}}(v,u)_i = 0$ for each i. Thus, d_G^{SPD} is a function of d_G^{RW}, and hence d_G^{RW} refines d_G^{SPD}. We finish by using Lemma 2 of Bevilacqua et al. (2022), which says that taking multisets of colors preserves

refinement. This shows that GD-WL with RRWP is at least as strong as GD-WL with SPD.

To show that GD-WL with RRWP is strictly stronger, we present a pair of non-isomorphic graphs that it can distinguish but GD-WL with SPD cannot: the Desargues graph and the Dodecahedral graph, which are plotted in Figure 6. Zhang et al. (2023) note that GD-WL with shortest path distances cannot distinguish these graphs. However, GD-WL with our RRWP positional encoding can. This can be seen from the 5-hop random walk probability distributions — there exists at least one walk of exactly length 5 between any two nodes of the Dodecahedral graph, but there is no exactly length 5 walk between many pairs of nodes of the Desargues graph. □

C.2. RRWP and MLP

Proof of Proposition 3.1. (a) First, we construct an MLP such that $\mathrm{MLP}(\mathbf{P})_{i,j} \approx \mathrm{SPD}_{K-1}(i,j)$, in which $\mathrm{SPD}_{K-1}(i,j)$ takes the value of the length of the shortest path between nodes i and j if i and j are no more than $K-1$ hops away from each other, and takes the value of K otherwise.

We build the MLP to approximate a composition of several continuous functions $f_3 \circ f_2 \circ f_1$. Let $\mathbf{P}(\mathbf{A})$ be the RRWP associated to the adjacency matrix $\mathbf{A}$, i.e. $\mathbf{P}(\mathbf{A}) = [\mathbf{I}, \mathbf{D}^{-1}\mathbf{A}, \ldots, (\mathbf{D}^{-1}\mathbf{A})^{K-1}]$. We define L to be a lower bound on the smallest nonzero entry of $\mathbf{P}(\mathbf{A})$, across all $\mathbf{A} \in \mathbb{G}_n$. In particular, we let

$$L = \min_{\mathbf{A}\in\mathbb{G}_n} \min_{i,j,t:\mathbf{P}(\mathbf{A})_{ijt}>0} \mathbf{P}(\mathbf{A})_{ijt}. \tag{7}$$

Since the minimizations are over finitely many positive entries, we have that $L > 0$. Thus, there exists a continuous function $f_1 : \mathbb{R}^K \to \mathbb{R}^K$ such that $f_1(x)_i = 0$ if $x_i \leq 0$ and 1 if $x_i \geq L$. Since for $t \in \{0, \ldots, K-1\}$ $\mathbf{P}_{ijt} \geq L$ if and only if node i is reachable to j in t hops, we have that

$$f_1(\mathbf{P}_{i,j,:})_t = \begin{cases} 1 & \text{if } i \text{ can reach } j \text{ in } t \text{ hops} \\ 0 & \text{else} \end{cases}. \tag{8}$$

Next, we define $f_2 : \mathbb{R}^K \to \mathbb{R}^K$ as $f_2(x)_t = \max_{t' \leq t} x_{t'}$. Then we have

$$(f_2 \circ f_1(\mathbf{P}_{i,j,:}))_t = \begin{cases} 1 & \text{if } \mathrm{SPD}(i,j) \leq t \\ 0 & \text{else} \end{cases}. \tag{9}$$

Finally, we let $f_3 : \mathbb{R}^K \to \mathbb{R}$ be defined by $f_3(x) = n - \sum_{t=1}^{K-1} x_t$. The full composition then gives

$$f_3 \circ f_2 \circ f_1(\mathbf{P}_{i,j,:}) = \begin{cases} \mathrm{SPD}(i,j) & \text{if } \mathrm{SPD}(i,j) \leq K-1 \\ n & \text{else} \end{cases}. \tag{10}$$

Now, note that $f_3 \circ f_2 \circ f_1$ is continuous, so by standard universal approximation results (Hornik et al., 1989) we can approximate it with an MLP on the compact set $\mathbb{G}_n$ to an arbitrary accuracy $\epsilon > 0$. This finishes the proof.

(b) Now, fix $\theta_0, \ldots, \theta_{K-1} \in \mathbb{R}$. We will construct an MLP such that $\mathrm{MLP}(\mathbf{P}) \approx \sum_{k=0}^{K-1} \theta_k (\mathbf{D}^{-1}\mathbf{A})^k$. Note that this target function can be written as

$$\sum_{k=0}^{K-1} \theta_k \mathbf{M}^k = \sum_{k=0}^{K-1} \theta_k \mathbf{P}_{:,:,k}. \tag{11}$$

Hence, we let $f_1 : \mathbb{R}^K \to \mathbb{R}^k$ be the continuous function that scales a vector $x \in \mathbb{R}^K$ elementwise by $\boldsymbol{\theta} = [\theta_0, \ldots, \theta_{K-1}]$: $f_1(x) = x \odot \boldsymbol{\theta}$. Then let $f_2 : \mathbb{R}^K \to \mathbb{R}$ take the sum: $f_2(x) = \sum_{k=0}^{K-1} x_k$. It is easy to see that

$$f_2 \circ f_1(\mathbf{P}_{i,j}) = \sum_{k=0}^{K-1} \theta_k \mathbf{P}_{i,j,k}, \tag{12}$$

so letting the MLP approximate the continuous function $f_2 \circ f_1$, we are done.

(c) Finally, we will construct an MLP such that $\mathrm{MLP}(\mathbf{P}) \approx \theta_0 \mathbf{I} + \theta_1 \mathbf{A}$. Note that this target function is equal to $\theta_0 \mathbf{P}_{:,:,0} + \theta_1 \mathbf{A}$.

To get the adjacency $\mathbf{A}$, we use the function $f_1 : \mathbb{R}^K \to \mathbb{R}^K$ as used in the proof of part (a), which rounds $f_1(x)_i = 0$ if $x_i \leq 0$ and 1 if $x_i \geq L$. Then $f_1(\mathbf{P}_{i,j})_1 = \mathbf{A}_{i,j}$. Then we take $f_2 : \mathbb{R}^K \to \mathbb{R}^K$ to scale the first two entries by θ_0 and θ_1 respectively (similarly to f_1 in the proof of part (b)), and take $f_3 : \mathbb{R}^K \to \mathbb{R}$ to sum across the first two slices: $f_3(x) = x_0 + x_1$. Then we have that

$$f_3 \circ f_2 \circ f_1(\mathbf{P}_{i,j}) = \theta_0 \mathbf{I} + \theta_1 \mathbf{A} \tag{13}$$

as desired. Choosing an MLP that approximate this continuous function $f_3 \circ f_2 \circ f_1$, we are done. □

C.3. LayerNorm on Nodes Removes the Degree Information from Sum-Aggregators and/or Degree Scalers

Normalization layers are essential for deep neural networks, especially Transformers (Vaswani et al., 2017). Graph Transformers usually use BatchNorm (Ioffe & Szegedy, 2015), following most MPNNs, or LayerNorm (Ba et al., 2016), following Transformers in other domains (Vaswani et al., 2017; Dosovitskiy et al., 2020). Here, we show that a LayerNorm removes degree information, which motivates our choice of BatchNorm in our Transformer.

Proof of Proposition 3.3. As noted by Corso et al. (2020), the output representation for a node i from a sum-aggregator can be viewed as $\mathbf{x}_i^{\text{sum}} = d_i \cdot \mathbf{x}_i^{\text{mean}}$ where $d_i \in \mathbb{R}$ is the degree of node i and $\mathbf{x}_i^{\text{mean}} = [x_{i1}, \dots x_{iF}]^\top \in \mathbb{R}^F$ is the node representation from a mean-aggregator. The layer normalization statistics for a node i over all hidden units are computed as follows:

$$\begin{aligned}
\mu_i^{\text{sum}} &= \frac{1}{F}\sum_{j=1}^{F} x_{ij}^{\text{sum}} = \frac{1}{F}\sum_{j=1}^{F} d_i \cdot x_{ij}^{\text{mean}} = \frac{d_i}{F}\sum_{j=1}^{F} x_{ij}^{\text{mean}} = d_i \cdot \mu_i^{\text{mean}} \\
\sigma_i^{\text{sum}} &= \sqrt{\frac{1}{F}\sum_{j=1}^{F}(x_{ij}^{\text{sum}} - \mu^{\text{sum}})^2} = \sqrt{\frac{d_i^2}{F}\sum_{j=1}^{F}(x_{ij}^{\text{mean}} - \mu^{\text{mean}})^2} = d_i \cdot \sigma_i^{\text{mean}}
\end{aligned} \tag{14}$$

Therefore, regardless of the elementwise affine transforms shared by all nodes, each element of the normalized representation

$$\tilde{x}_{ij}^{\text{sum}} = \frac{(x_{ij}^{\text{sum}} - \mu_i^{\text{sum}})}{\sigma_i^{\text{sum}}} = \frac{(d_i \cdot x_{ij}^{\text{mean}} - d_i \cdot \mu_i^{\text{mean}})}{d_i \cdot \sigma_i^{\text{mean}}} = \frac{(x_{ij}^{\text{mean}} - \mu_i^{\text{mean}})}{\sigma_i^{\text{mean}}} = \tilde{x}_{ij}^{\text{mean}}, \quad \forall i \in \mathcal{V}, \forall j = 1, \dots, F, \tag{15}$$

will be the same for both sum-aggregation and mean-aggregation.

The same conclusion can be seen for degree scalers, by simply changing d_i to $f(d_i)$ in the proof, where $f : \mathbb{R} \to \mathbb{R}_{>0}$.

□