CMSC 35430: Exploring Graph Transformers – From Core Architectures to Positional Encoding Strategies

Dongwei Lyu
dwlyu@uchicago.edu

March 19, 2025

Abstract

GraphGPS provides a flexible framework for integrating various positional encodings and attention mechanisms, achieving state-of-the-art performance across diverse graph benchmarks. This paper begins with an in-depth exploration of the structure and theoretical foundations of GraphGPS, focusing on key architectural components such as message-passing networks, including Gated Convolutional Networks (GCN) and Graph Isomorphism Networks with Edge Features (GINE), as well as positional encodings like Learnable Laplacian Positional Encoding (LaPE) and Random Walk Structural Embeddings (RWSE). Additionally, we examine spectral-based graph Transformers that operate without message-passing layers. Given the strong performance of GRIT, which incorporates novel positional encoding strategies, we further investigate the role of relative random walk positional encoding by experimenting with Relative Random Walk Probabilities (RRWP) within the Spectral Attention Network (SAN). Our empirical evaluation reveals that GRIT achieves the best overall performance, while SAN with RWSE performs the best among all SAN configurations. Furthermore, our analysis indicates that LaPE is not always the optimal choice for SAN, and alternative random-walk-based positional encodings can offer competitive results. While RRWP alone performs poorly, it enhances SAN's performance when combined with LaPE, underscoring the importance of absolute positional embeddings. Additionally, our findings confirm the strong coupling between RRWP and SAN's attention mechanism in GRIT, reinforcing the effectiveness of this combination. These insights highlight the need for careful selection of positional encodings based on model architecture and dataset characteristics, providing valuable guidance for advancing graph representation learning. Code Available at `https://github.com/dwlyu/graph_gps_pos`.

1 Introduction

Graph-structured data are ubiquitous in domains such as social networks, chemistry, and knowledge graphs. Graph Neural Networks (GNNs) based on message passing have become a standard tool to learn representations from such data by iteratively aggregating information from local neighborhoods. While message-passing GNNs have achieved strong results, they face fundamental limitations, including over-smoothing (node representations becoming indistinguishably similar after many layers), over-squashing (exponential neighborhood information compressed into fixed-size vectors), and restricted expressive power under the 1-Weisfeiler–Lehman graph isomorphism test. These issues arise because traditional GNNs only allow information to propagate locally, making it hard to capture long-range dependencies without deep stacks of layers. To address these shortcomings, Graph Transformers (GTs) have recently emerged. Graph Transformers generalize the Transformer architecture to graphs by allowing each node to attend to all other nodes (global self-attention), thereby directly incorporating long-range interactions. By doing so, Graph Transformers can alleviate the bottlenecks of purely local message passing, mitigating over-smoothing and over-squashing and potentially exceeding the expressiveness limits of message-passing GNNs.

Early attempts to bring Transformers to graphs needed to overcome the lack of a natural sequence or grid structure in graph data. In a standard Transformer, positional encodings are added to tokens to convey their order or coordinates. For graphs, there is no canonical ordering or coordinate system for nodes, so one must introduce positional or structural encodings that reflect the graph's topology. The first Graph Transformer, proposed by Dwivedi & Bresson (2020), addressed this by using the graph Laplacian's eigenvectors as node positional encodings. Subsequently, more graph transformers have been developed, incorporating various strategies to encode graph structures or enhance the attention mechanism. Notable examples include:

- Graphormer by Ying et al. (2021) takes a different approach, encoding pairwise structural information as learnable biases in attention. It leverages shortest path distances (or 3D distances in molecular graphs) as a form of relative positional encoding, enabling the model to capture graph connectivity patterns without explicit message passing. Graphormer achieved outstanding success on large-scale molecular property prediction tasks, showcasing the power of fully-connected attention with well-chosen positional encodings.

- GraphiT (Mialon et al., 2021) introduces diffusion kernel-based encodings: it uses the heat diffusion information of the graph to modulate attention scores between nodes. By incorporating these diffusion-based relative positional encodings, GraphiT can bias the Transformer to respect graph structure, even though the attention is global.

- GraphTrans (Jain et al., 2021) represents a hybrid strategy: it stacks a few layers of standard message-passing GNN (to extract local patterns) and then applies a Transformer layer over all nodes to capture long-range context. This two-stage design (local GNN + global Transformer) was one of the first to explicitly combine message passing with global attention in a unified model, which demonstrated outstanding performance on long range tasks.

2 Motivation

For this final project, I aim to develop a systematic understanding of graph transformers but found that the field encompasses a highly diverse range of models. However, I came across a foundational paper introducing a unified framework for scalable graph transformers called GraphGPS. GraphGPS provides a structured recipe for building Graph Transformers by integrating three key components as shown in Figure 1:

- positional/structural encodings
- local message-passing mechanism
- global attention mechanism

In essence, GraphGPS interleaves efficient **message-passing layers** (operating on the actual graph edges) with Transformer layers that implement **global self-attention** among all nodes. By decoupling local aggregation from global fully-connected attention, GraphGPS effectively captures local neighbor structures, providing a strong inductive bias, while global attention enables long-range connectivity. This design preserves the expressivity of full self-attention—making the model a universal function approximator on graphs with sufficiently rich encodings—while scaling to much larger graphs than previous fully-connected Graph Transformers. The GraphGPS framework provides a modular template for integrating various positional encodings and attention mechanisms, achieving state-of-the-art results across diverse graph benchmarks. Therefore, I chose it as a starting point to explore how different models function by examining the key components of the GraphGPS architecture in depth.

Project Outline: In this paper, we primarily focus on understanding the core message-passing network modules from the original GraphGPS papers, including the **Gated Convolutional Network (GCN), Graph Isomorphism Network with Edge Features (GINE)**, as well as different positional encodings such as **Learnable Laplacian Positional Encoding (LaPE)** and **Random Walk Structural Embeddings (RWSE)**.

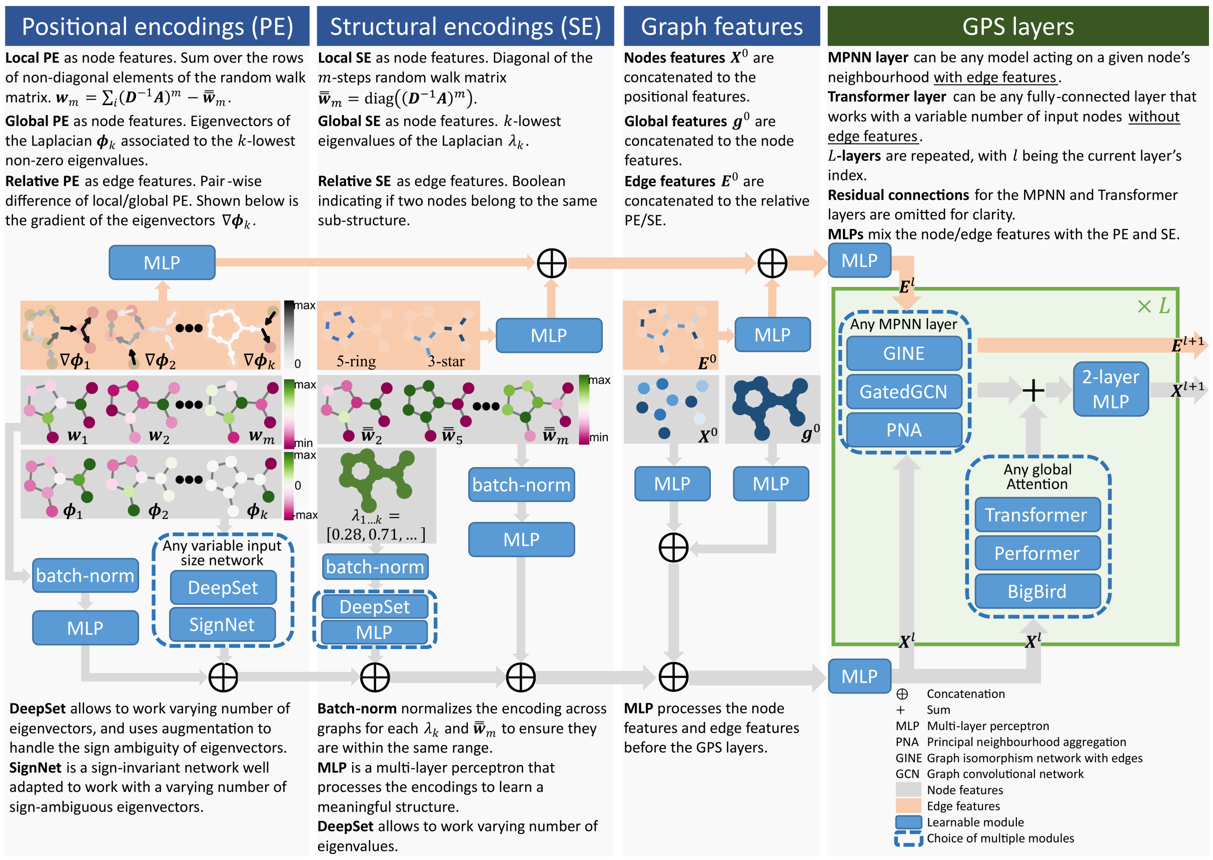

Positional encodings (PE)
Local PE as node features. Sum over the rows of non-diagonal elements of the random walk matrix. $w_m = \sum_i (D^{-1}A)^m - \bar{\bar{w}}_m$.
Global PE as node features. Eigenvectors of the Laplacian ϕ_k associated to the k-lowest non-zero eigenvalues.
Relative PE as edge features. Pair-wise difference of local/global PE. Shown below is the gradient of the eigenvectors $\nabla\phi_k$.
Structural encodings (SE)
Local SE as node features. Diagonal of the m-steps random walk matrix $\bar{\bar{w}}_m = \mathrm{diag}((D^{-1}A)^m)$.
Global SE as node features. k-lowest eigenvalues of the Laplacian λ_k.
Relative SE as edge features. Boolean indicating if two nodes belong to the same sub-structure.
Graph features
Nodes features X^0 are concatenated to the positional features.
Global features g^0 are concatenated to the node features.
Edge features E^0 are concatenated to the relative PE/SE.
GPS layers
MPNN layer can be any model acting on a given node's neighbourhood with edge features.
Transformer layer can be any fully-connected layer that works with a variable number of input nodes without edge features.
L-layers are repeated, with l being the current layer's index.
Residual connections for the MPNN and Transformer layers are omitted for clarity.
MLPs mix the node/edge features with the PE and SE.
MLP
$\nabla\phi_1$ $\nabla\phi_2$ $\nabla\phi_k$
max 0 min
w_1 w_2 w_m
ϕ_1 ϕ_2 ϕ_k
5-ring 3-star
$\bar{\bar{w}}_2$ $\bar{\bar{w}}_5$ $\bar{\bar{w}}_m$
$\lambda_{1...k} = [0.28, 0.71, ...]$
E^0 X^0 g^0
batch-norm
Any variable input size network
DeepSet
SignNet
E^l E^{l+1} X^l X^{l+1}
$\times L$
Any MPNN layer
GINE
GatedGCN
PNA
2-layer MLP
Any global Attention
Transformer
Performer
BigBird
DeepSet allows to work varying number of eigenvectors, and uses augmentation to handle the sign ambiguity of eigenvectors.
SignNet is a sign-invariant network well adapted to work with a varying number of sign-ambiguous eigenvectors.
Batch-norm normalizes the encoding across graphs for each λ_k and $\bar{\bar{w}}_m$ to ensure they are within the same range.
MLP is a multi-layer perceptron that processes the encodings to learn a meaningful structure.
DeepSet allows to work varying number of eigenvalues.
MLP processes the node features and edge features before the GPS layers.
$\oplus$ Concatenation
$+$ Sum
MLP Multi-layer perceptron
PNA Principal neighbourhood aggregation
GINE Graph isomorphism network with edges
GCN Graph convolutional network
Node features
Edge features
Learnable module
Choice of multiple modules

Figure 1: Illustration of the GraphGPS architecture

To further deepen my understanding of architectural choices in graph Transformers, we then explore recent advances in spectral-based graph Transformers without message-passing layers.

2.1 Graph Transformers without Message Passing

However, integrating MPNNs with Transformers increases model complexity and training difficulty, necessitating careful hyperparameter tuning to balance message passing, attention mechanisms, and positional encoding. To address these challenges, recent research has explored purely Transformer-based Graph Neural Networks, aiming to eliminate message passing while preserving strong structural inductive biases.

The **Spectral Attention Network (SAN)**is a fully-connected graph Transformer that completely removes message passing by leveraging **learned positional encodings (LPEs)** based on the graph Laplacian spectrum. Instead of iteratively aggregating neighborhood information, SAN encodes global structural relationships using eigenvectors of the Laplacian, which serve as spectral positional encodings. These LPEs are then added to node embeddings and passed through a standard Transformer architecture, where **full self-attention** enables direct interaction between all nodes, irrespective of the original graph connectivity. By removing the constraints of localized aggregation, SAN avoids the **over-squashing problem** commonly found in message-passing GNNs and achieves improved long-range dependency modeling.

A more recent model is the Graph Inductive Bias Transformer (GRIT), which also eliminates message passing while incorporating meaningful graph structure through learned **Relative Random Walk Probabilities (RRWP)**. Instead of iteratively aggregating neighborhood information, RRWP encodes relational graph properties by initializing node-pair interactions based on random walk transition probabilities, ensuring that structural relationships are captured without explicit message exchange. This design fundamentally differentiates GRIT from MPNN-based methods, replacing iterative propagation with a learned representation that implicitly encodes structural connectivity.

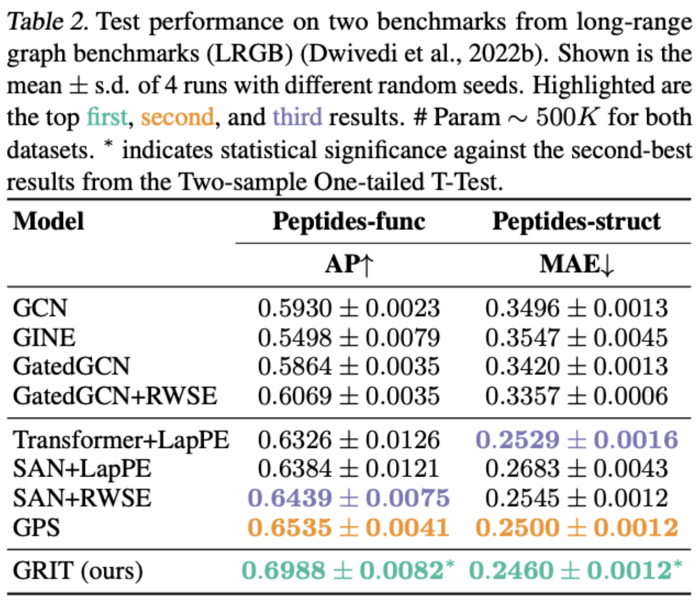


Table 2. Test performance on two benchmarks from long-range graph benchmarks (LRGB) (Dwivedi et al., 2022b). Shown is the mean ± s.d. of 4 runs with different random seeds. Highlighted are the top first, second, and third results. # Param $\sim 500K$ for both datasets. * indicates statistical significance 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)	**$0.6988 \pm 0.0082^*$**	**$0.2460 \pm 0.0012^*$**

Figure 2: GRIT accuracy with different positional embedding strategy

Beyond positional encodings, GRIT introduces a **dual-update Transformer architecture**, where attention is applied to both **node embeddings** and to **node-pair representations**, allowing the model to dynamically refine both individual node states and their relational interactions. To maintain local structural properties, GRIT further incorporates degree information into attention updates and replaces standard layer normalization with batch normalization, ensuring that degree-based features remain meaningful throughout training. This architecture enables GRIT to leverage global attention while avoiding message-passing bottlenecks, making it particularly well-suited for large-scale graph learning tasks.

Project Outline: GRIT achieves SOTA performance on several benchmark datasets with its two key innovations. In their paper, the authors conducted ablation studies on the impact of different positional encoding strategies, as shown in the Figure 2. However, I am particularly curious about the extent to which the attention mechanism contributes to this performance. Therefore, I aim to experiment with the **Relative Random Walk Probabilities (RRWP)** encoding strategy within the **Spectral Attention Network (SAN)** to evaluate its effectiveness on benchmark datasets and gain deeper insights into its impact.

3 Message Passing Graph Neural Network

3.1 Gated Graph Convolutional Neural Network

Gated Graph Convolutional Neural Networks (GCNs) extend traditional convolutional neural networks to graph-structured data, enabling effective representation learning on non-Euclidean domains. Unlike standard CNNs, which operate on regular grid structures (e.g., images), GCNs leverage the inherent connectivity of graphs to propagate and aggregate information across nodes.

A graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the set of N nodes, and $\mathcal{E}$ is the set of edges. Each node $v \in \mathcal{V}$ has a feature vector x_v, and edges $(u, v) \in \mathcal{E}$ may have associated weights or features c_{uv}. The Gated GCN follows a **layer-wise message**

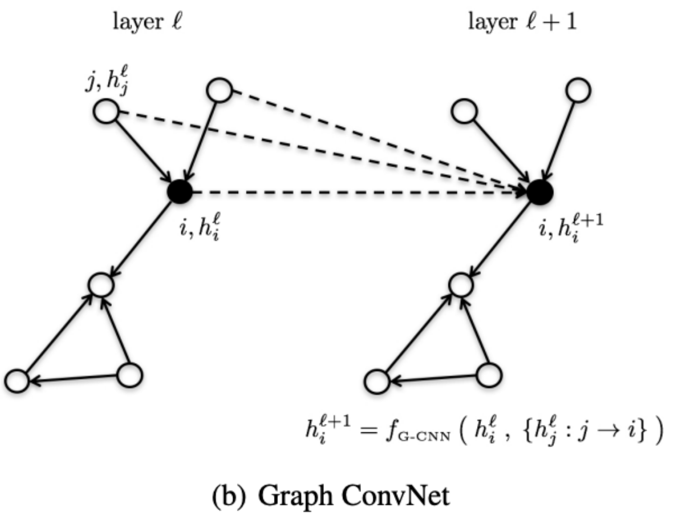


Figure 3: Message passing flow between node features for a graph ConvNet

passing framework where each node aggregates information from its neighbors as shown in Figure 3. Given the input node-wise feature as x and the updated node-wise feature as h for each layer, the most general formulation of the message-passing flow framework is:

$$h_v = \phi\left(x_u, \bigoplus_{v \in \mathcal{N}_u} c_{uv}\psi(x_v)\right) \tag{1}$$

where $\psi(\cdot)$ and $\phi(\cdot)$ are transformation functions applied to node features, c_{uv} represents the weight associated with edge (u, v), $\mathcal{N}_u$ denotes the set of neighbors of node u, and $\bigoplus$ is an aggregation operator.

In the specific case of Residual Gated Graph Convolutional Networks, the update function takes the form:

$$h_u = W_1 x_u + \sum_{v \in \mathcal{N}_u} \eta_{uv} \odot W_2 x_v \tag{2}$$

where W_1 and W_2 are trainable weight matrices, η_{uv} is a gating function controlling message flow, and $\odot$ represents element-wise multiplication.

The gating operation η_{uv} is defined as:

$$\eta_{uv} = \sigma(W_3 x_u + W_4 x_v) \tag{3}$$

where W_3 and W_4 are learnable parameters, $\sigma(\cdot)$ is a sigmoid activation function.

This formulation explicitly incorporates edge attributes to modulate message passing, unlike standard GCNs that use adjacency-based propagation. The gating mechanism η_{uv} refines node representations by adaptively weighting contributions from neighbors, preventing over-smoothing in deep GNNs. Furthermore, the residual connection in the update rule stabilizes training and allows deeper networks to propagate information efficiently.

3.2 Graph Isomorphism Network with Edge Features

Traditional Graph Convolutional Networks (GCNs) primarily rely on spectral-based aggregation, where node representations are updated by applying linear transformations to the neighborhood features, often using adjacency-based normalization. While effective in many applications, GCNs struggle with distinguishing different graph structures, as their aggregation functions are not injective over multisets of node features. This limitation restricts their ability to fully capture graph isomorphism properties. Graph Isomorphism Network (GIN) introduces **MLP layer** to replace the **weighted layer** for message aggregation.

Graph Isomorphism Network with Edge Features (GINE) extends GIN by integrating edge features into message passing, improving expressivity and structural awareness. Unlike traditional message-passing neural networks, which rely solely on node features, GINE refines information propagation by incorporating edge attributes, making it particularly effective for molecular property prediction and social network analysis.

3.2.1 Network Structure

The GINE message-passing update is defined as follows:

$$a_i^{(\ell)} = \sum_{j \in \mathcal{N}(i)} \sigma\left(h_j^{(\ell-1)} + E(e_{ij})\right) \tag{4}$$

$$h_i^{(\ell)} = \text{MLP}\left((1+\epsilon)h_i^{(\ell-1)} + a_i^{(\ell)}\right) \tag{5}$$

where:

- $a_i^{(\ell)}$ is the aggregated information
- $h_i^{(\ell)}$ is the hidden state of node i at layer ℓ,
- $\mathcal{N}(i)$ is the set of neighbors of node i,
- $E(e_{ij})$ is an embedding function for the edge feature between nodes i and j,
- $\sigma(\cdot)$ is a nonlinearity such as ReLU,
- **MLP** is a multi-layer perceptron for feature transformation,
- ϵ is a trainable parameter that enhances expressiveness.

This formulation extends GIN by incorporating **edge attributes**, enabling a richer representation of relational data. The function $E(e_{ij})$ maps edge labels into an embedding space, allowing edge-aware information propagation.

3.2.2 Virtual Node Mechanism

While standard MPNNs aggregate local node neighborhoods, they struggle with **long-range dependencies** due to limited receptive fields. To address this, they introduce a **virtual node** connected to all nodes in the graph, enabling efficient information exchange across distant nodes.

The virtual node aggregation step in GINE follows:

$$H^{(\ell)} = \text{MLP}\left((1+\epsilon)H^{(\ell-1)} + \sum_{i \in \mathcal{V}} \bar{h}_i^{(\ell)}\right) \tag{6}$$

$$h_i^{(\ell)} = \bar{h}_i^{(\ell)} + H^{(\ell)} \tag{7}$$

where:

- $H^{(\ell)}$ is the **virtual node embedding** at layer ℓ,
- $\bar{h}_i^{(\ell)}$ is the node embedding **before the virtual node update**,
- $\sum_{i \in \mathcal{V}} \bar{h}_i^{(\ell)}$ aggregates all node embeddings,

- The **MLP** applies transformation and nonlinearity to the virtual node,
- The final update integrates the virtual node state into each node's hidden representation.

By introducing a virtual node, GINE captures **global structural information** at every step while preserving the local **injective aggregation** properties of GIN. This allows for better generalization in tasks requiring **long-range dependencies**.

3.3 GPS Layer Definition

Given an albitraty message passing block as Gated GCN and GINE described above, at each layer, the features are updated by aggregating the output of an MPNN layer with that of a global attention layer. Note that the edge features are only passed to the MPNN layer, and that residual connections with batch normalization are omitted for clarity. Both the MPNN and GlobalAttn layers are **modular**, i.e., MPNN can be any function that acts on a local neighborhood and GlobalAttn can be any fully connected layer.

$$X^{(\ell+1)}, E^{(\ell+1)} = \mathrm{GPS}^{(\ell)}(X^{(\ell)}, E^{(\ell)}, A), \tag{8}$$

computed as

$$X_M^{(\ell+1)}, E^{(\ell+1)} = \mathrm{MPNN}_e^{(\ell)}(X^{(\ell)}, E^{(\ell)}, A), \tag{9}$$

$$X_T^{(\ell+1)} = \mathrm{GlobalAttn}^{(\ell)}(X^{(\ell)}), \tag{10}$$

$$X^{(\ell+1)} = \mathrm{MLP}^{(\ell)}(X_M^{(\ell+1)} + X_T^{(\ell+1)}). \tag{11}$$

4 Positional Encoding

Graph Transformers rely on positional encodings to address the lack of inherent ordering in graph data. Unlike sequence models, where order is naturally defined, graphs require specialized encodings to capture both relational structure and positional context. In this project, we explore two prominent techniques for graph-based positional encoding: spectral-based **Learnable Laplacian Positional Encoding (LaPE)** and random-walk based **Random Walk Structural Encoding (RWSE)**.

4.1 Learnable Laplacian Positional Encoding

4.1.1 Node-Based LaPE

Graph Laplacian and Eigenfunction Decomposition Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes, we define the normalized graph Laplacian as:

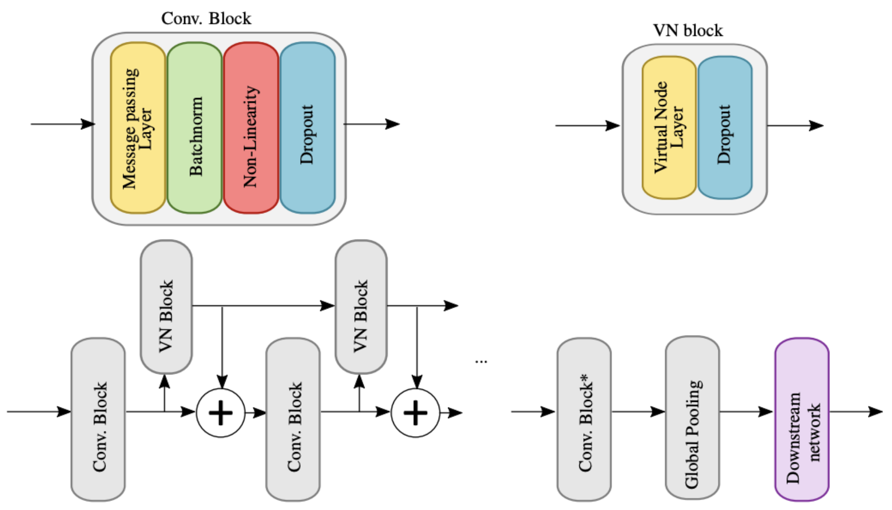


Figure 4: General structure of the network used in this work. Note that the Virtual node blocks are optional. Also note that the nonlinearity is usually omitted in the last Conv Block.

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \tag{12}$$

where $\mathbf{A}$ is the adjacency matrix, and $\mathbf{D}$ is the degree matrix.

The eigenvalues and eigenvectors of $\mathbf{L}$ capture fundamental structural properties of the graph. The eigen-decomposition of $\mathbf{L}$ yields:

$$\mathbf{L\Phi} = \mathbf{\Phi}\Lambda \tag{13}$$

where:

- $\mathbf{\Phi} = [\phi_1, \phi_2, \ldots, \phi_m] \in \mathbb{R}^{N\times m}$ contains the first m eigenvectors (sorted by increasing eigenvalue),
- $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m)$ is the diagonal matrix of corresponding eigenvalues.

These eigenvectors encode node positional information in a globally consistent manner, capturing both local neighborhoods and global graph topology. The node-level positional encoding assigns each node i a feature vector based on the corresponding row in $\mathbf{\Phi}$, i.e.,

$$p_i^{(\mathrm{LaPE})} = [\phi_{i,1}, \phi_{i,2}, \ldots, \phi_{i,m}] \tag{14}$$

where $\phi_{i,j}$ is the j-th element of the eigenvector associated with the i-th lowest eigenvalue. These eigenvectors encode each node's position relative to the entire graph structure, capturing both local neighborhood information and global topology.

As shown in Figure 5, the initial positional encodings $p^{(\mathrm{LaPE})}$ can be then turn into

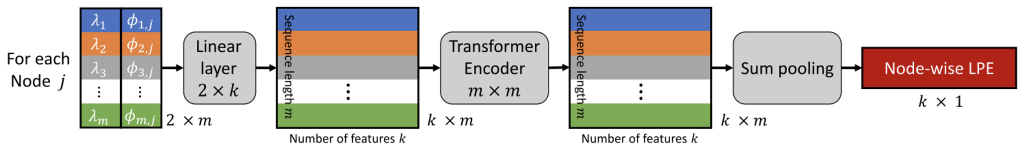


Figure 5: **Learned Positional Encoding (LPE) Architecture:** The model incorporates the graph's Laplace spectrum by utilizing the first m eigenvalues and eigenvectors, where $m \leq N$, with N representing the number of nodes. The Transformer processes nodes in parallel, treating each node as an element of a batch. Here, $\phi_{i,j}$ denotes the j-th component of the eigenvector corresponding to the i-th lowest eigenvalue λ_i.

learnable feature vectors as input via projector:

$$p_i' = LL_p(p_i^{(\text{LaPE})}) = C_0 p_i^{(\text{LaPE})} + c_0 \in \mathbb{R}^d, \tag{15}$$

4.1.2 Edge-Based LaPE

While node-based LaPE captures absolute spectral positions of nodes, it does not explicitly encode relative positional relationships between pairs of nodes. To address this, we introduce an edge-based LaPE formulation that leverages **sign-invariant spectral differences**.

For an edge e_{uv} connecting nodes u and v, we compute two sign-invariant spectral operators:

$$\mathbf{E}_{uv}^{(1)} = |\mathbf{\Phi}_{u,1:m} - \mathbf{\Phi}_{v,1:m}| \tag{16}$$

$$\mathbf{E}_{uv}^{(2)} = \mathbf{\Phi}_{u,1:m} \odot \mathbf{\Phi}_{v,1:m} \tag{17}$$

where:

- $\mathbf{E}_{uv}^{(1)}$ captures absolute spectral differences, ensuring robustness to eigenvector sign ambiguity.
- $\mathbf{E}_{uv}^{(2)}$ encodes frequency alignment, measuring spectral similarity.

4.2 random walk structure encoding

Definition

Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes, the **random walk operator** is defined as:

$$\mathbf{RWP} = \mathbf{AD}^{-1} \tag{18}$$

where:

- $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix,
- $\mathbf{D} \in \mathbb{R}^{N \times N}$ is the diagonal degree matrix with $D_{ii} = \sum_j A_{ij}$.

The **k-step random walk structure encoding** for node i is then defined as:

$$p_i^{(\text{RWSE})} = \left[(\mathbf{RW})_{ii}, (\mathbf{RW}^2)_{ii}, \dots, (\mathbf{RW}^k)_{ii} \right] \in \mathbb{R}^k \tag{19}$$

where $(\mathbf{RW}^t)_{ii}$ represents the probability that a random walker starting at node i returns to i after t steps.

Unlike LaPE, RWSE is inherently sign-invariant and does not require additional preprocessing for consistency. However, RWSE's ability to differentiate nodes depends on the uniqueness of their k-hop neighborhoods; for sufficiently large k, it provides a distinct node representation in most graphs.

In highly regular graphs, such as CSL graphs, RWSE assigns the same encoding to all nodes due to their structural equivalence, whereas LaPE may still retain some distinguishing features through eigenvector variations.

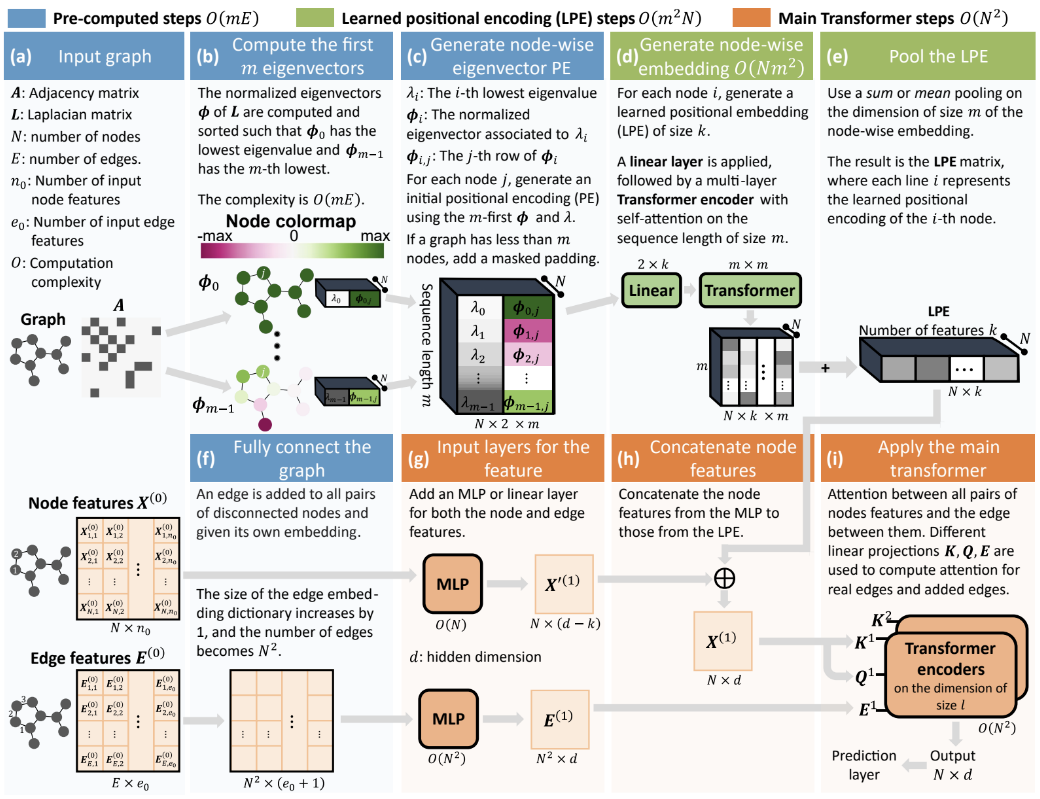
Despite this, RWSE uniquely classifies different classes of isomorphic graphs, leading to perfect classification in synthetic benchmarks such as CSL datasets. RWSE serves as an effective alternative to spectral-based positional encodings, balancing expressivity, computational efficiency, and robustness to sign ambiguity.

5 Spectral-based Transformers

In this section, we introduce purely Transformer-based Graph Neural Networks designed to eliminate message passing while retaining strong structural inductive biases. One of the most fundamental Transformer architectures without message passing layers is the **Graph Transformer with Spectral Attention (SAN)**, which was introduced in the same work that proposed the **Laplacian Positional Encoding (LaPE)** discussed earlier. In the following, we will explore its attention mechanism in detail.

5.1 Attention Mechanism in SAN

The attention mechanism in the Transformer extends the standard self-attention mechanism by incorporating **graph structure** and **edge features**, enabling the model to capture both local connectivity and global dependencies. This approach preserves the flexibility of fully connected self-attention while maintaining structural inductive biases through edge-aware attention mechanisms.



The normalized eigenvectors ϕ of L are computed and sorted such that ϕ_0 has the lowest eigenvalue and ϕ_{m-1} has the m-th lowest.

The complexity is $O(mE)$.

λ_i: The i-th lowest eigenvalue
ϕ_i: The normalized eigenvector associated to λ_i
$\phi_{i,j}$: The j-th row of ϕ_i
For each node j, generate an initial positional encoding (PE) using the m-first ϕ and λ.
If a graph has less than m nodes, add a masked padding.

For each node i, generate a learned positional embedding (LPE) of size k.

A **linear layer** is applied, followed by a multi-layer **Transformer encoder** with self-attention on the sequence length of size m.

Use a *sum* or *mean* pooling on the dimension of size m of the node-wise embedding.

The result is the **LPE** matrix, where each line i represents the learned positional encoding of the i-th node.

An edge is added to all pairs of disconnected nodes and given its own embedding.

The size of the edge embedding dictionary increases by 1, and the number of edges becomes N^2.

Add an MLP or linear layer for both the node and edge features.

d: hidden dimension

Concatenate the node features from the MLP to those from the LPE.

Attention between all pairs of nodes features and the edge between them. Different linear projections K, Q, E are used to compute attention for real edges and added edges.

Figure 6: Overview of the architecture for Graph Transformer with Spectral Attention (SAN)

5.1.1 Multi-Head Attention with Edge Features

Let $h_i^{(l)}$ denote the feature vector of node i at layer l, and let e_{ij} represent the edge feature embedding for the edge between nodes i and j. The Transformer employs a **multi-head attention mechanism** over all nodes, defined as:

$$\hat{h}_i^{(l+1)} = O_h^{(l)} \left[{}_{k=1}^{H} \sum_{j \in \mathcal{V}} w_{ij}^{(k,l)} V^{(k,l)} h_j^{(l)} \right] \tag{20}$$

where:

- $O_h^{(l)} \in \mathbb{R}^{d \times d}$ is an output projection matrix.
- $V^{(k,l)} \in \mathbb{R}^{d_k \times d}$ is the value transformation matrix for head k.
- H denotes the number of attention heads, and represents concatenation.
- d is the hidden dimension, and $d_k = \frac{d}{H}$ is the dimension of each head.

5.1.2 Graph-Aware Attention Weights

A key distinction in the Transformer is the **graph-aware attention mechanism**, where attention weights depend on both node and edge features. Specifically, separate attention mechanisms are applied for **connected** and **unconnected** node pairs, preserving both local graph connectivity and global interactions.

The unnormalized attention weights at layer l and head k are given by:

$$\hat{w}_{ij}^{(k,l)} = \begin{cases} \frac{Q_1^{(k,l)} h_i^{(l)} \circ K_1^{(k,l)} h_j^{(l)} \circ E_1^{(k,l)} e_{ij}}{\sqrt{d_k}}, & \text{if } (i,j) \in \mathcal{E} \text{ (connected nodes)} \\ \frac{Q_2^{(k,l)} h_i^{(l)} \circ K_2^{(k,l)} h_j^{(l)} \circ E_2^{(k,l)} e_{ij}}{\sqrt{d_k}}, & \text{otherwise} \end{cases} \tag{21}$$

where:

- $\circ$ denotes element-wise multiplication.
- $Q_1^{(k,l)}, K_1^{(k,l)}, E_1^{(k,l)} \in \mathbb{R}^{d_k \times d}$ are query, key, and edge projection matrices for **connected** node pairs.
- $Q_2^{(k,l)}, K_2^{(k,l)}, E_2^{(k,l)} \in \mathbb{R}^{d_k \times d}$ are query, key, and edge projection matrices for **unconnected** node pairs.

The final **normalized attention weights** are computed as:

$$w_{ij}^{(k,l)} = \begin{cases} \frac{1}{1+\gamma} \cdot \text{softmax}\left(\sum_{d_k} \hat{w}_{ij}^{(k,l)}\right), & \text{if } (i,j) \in \mathcal{E} \text{ (connected nodes)} \\ \frac{\gamma}{1+\gamma} \cdot \text{softmax}\left(\sum_{d_k} \hat{w}_{ij}^{(k,l)}\right), & \text{otherwise} \end{cases} \tag{22}$$

where $\gamma \in \mathbb{R}^+$ is a hyperparameter controlling the bias toward full-graph attention. This flexibility allows the model to adapt to datasets with varying levels of long-range dependencies.

Numerical Stability Considerations: The softmax outputs are clamped between -5 and 5 to prevent numerical instability, and distinct key, query, and edge projection matrices are used for connected and disconnected node pairs.

5.1.3 Node Representation Updates

Following the attention computation, a **multi-layer perceptron (MLP)** is applied with residual connections and normalization layers to refine node representations:

$$\tilde{h}_i^{(l+1)} = \text{Norm}\left(h_i^{(l)} + \hat{h}_i^{(l+1)}\right) \tag{23}$$

$$\tilde{\tilde{h}}_i^{(l+1)} = W_2^{(l)} \text{ReLU}\left(W_1^{(l)} \tilde{h}_i^{(l+1)}\right) \tag{24}$$

$$h_i^{(l+1)} = \text{Norm}\left(\tilde{h}_i^{(l+1)} + \tilde{\tilde{h}}_i^{(l+1)}\right) \tag{25}$$

where:

- $W_1^{(l)} \in \mathbb{R}^{2d \times d}$ and $W_2^{(l)} \in \mathbb{R}^{d \times 2d}$ are learnable weight matrices.

- Norm denotes layer normalization.
- Residual connections stabilize training and prevent information degradation.

5.2 Graph Inductive bias Transformer

Another recent advancement in developing purely Transformer-based architectures is the Graph Inductive Bias Transformer (GRIT). GRIT incorporates **Relative Random Walk Probabilities (RRWP)**, a positional encoding scheme designed to capture multi-hop dependencies and structural relationships between nodes. Additionally, GRIT introduces a novel **flexible attention mechanism** that updates both node and node-pair representations, enabling richer feature propagation and improved expressivity in graph learning tasks.

5.2.1 Relative Random Walk Probabilities (RRWP)

Relative Random Walk Probabilities (RRWP) is a positional encoding scheme designed to capture multi-hop dependencies and structural relationships between nodes in a graph. Unlike spectral-based positional encodings, such as Laplacian Eigenvectors, which rely on eigen-decomposition, RRWP leverages **random walk dynamics** to encode relative positional information while maintaining computational efficiency. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with $n = |\mathcal{V}|$ nodes. Define the **adjacency matrix** $\mathbf{A} \in \mathbb{R}^{n\times n}$ and the **diagonal degree matrix** $\mathbf{D}$, where $D_{ii} = \sum_j A_{ij}$. We introduce the **random walk transition matrix**:

$$\mathbf{M} = \mathbf{D}^{-1}\mathbf{A} \tag{26}$$

where M_{ij} represents the probability that node i transitions to node j in one step of a simple random walk. The Relative Random Walk Probabilities (RRWP) positional encoding is defined for each node pair $(i, j) \in \mathcal{V}$ as:

$$P_{i,j} = \left[\mathbf{I}, \mathbf{M}, \mathbf{M}^2, \ldots, \mathbf{M}^{K-1}\right]_{i,j} \in \mathbb{R}^K \tag{27}$$

where:

- $\mathbf{I}$ is the identity matrix, ensuring that self-position information is included.
- $\mathbf{M}^k$ denotes the k-step transition probability in a random walk.
- The parameter $K \in \mathbb{N}$ controls the maximum number of steps considered in the random walk.

For all the node $i \in \mathcal{V}$, the **diagonal elements** $P_{i,i}$ provide a node-specific encoding:

$$P_{i,i} = \left[1, (\mathbf{M})_{i,i}, (\mathbf{M}^2)_{i,i}, \ldots, (\mathbf{M}^{K-1})_{i,i}\right] \in \mathbb{R}^K \tag{28}$$

which is equivalent to the random walk structure encodings (RWSE) introduced in previous section. Relative Random Walk Probabilities extend RWSE by encoding not only self-return probabilities but also transition probabilities between distinct nodes, thereby preserving relative positional information in the graph. This makes RRWP more effective for capturing **multi-hop dependencies** and **structural similarities** between nodes. Furthermore, RWSE treats each node independently, limiting its ability to model node interactions, whereas RRWP explicitly incorporates pairwise relationships, making it better suited for attention-based architectures.

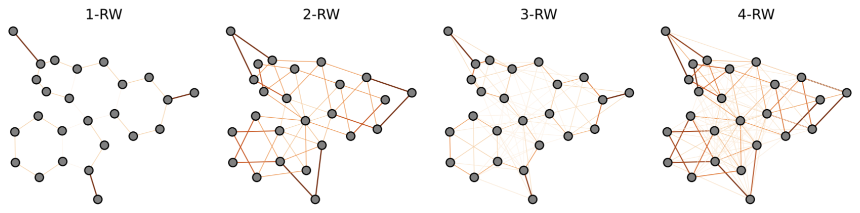


Figure 1. RRWP visualization for the fluorescein molecule, up to the 4th power. Thicker and darker edges indicate higher edge weight. Probabilities for longer random walks reveal higher-order structures (e.g., the cliques evident in 3-RW and the star patterns in 4-RW).

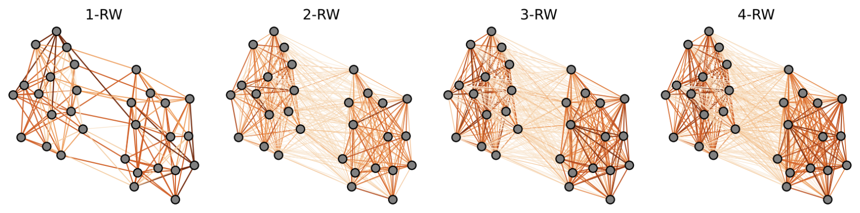


Figure 2. RRWP visualization for a sample from a stochastic block model with 2 communities, up to the 4th power. Probabilities for longer random walks better highlight the community structure and reduce bottlenecks.

Rather than treating RRWP as a fixed encoding, it is used as an **initialization** for learnable relative positional encodings within the Transformer architecture. The positional encoding tensor P is transformed by an element-wise Multi-Layer Perceptron (MLP):

$$P'_{i,j} = \text{MLP}(P_{i,j,:}) \in \mathbb{R}^d \tag{29}$$

where:

- $\text{MLP} : \mathbb{R}^K \to \mathbb{R}^d$ maps the initial RRWP encodings to a higher-dimensional learned representation.
- d is the hidden dimension used in the Transformer.

Additionally, the RRWP representations are updated dynamically within the **attention layers** of the Transformer, allowing the model to refine the positional information based on learned node interactions.

5.2.2 Theory: RRWP + MLP is Expressive

Given the initial RRWP positional encodings P, a MLP is applied to learn new positional encodings in an end-to-end manner. This combination is provably expressive, as the learned positional encoding can approximate shortest path distances and general classes of graph propagation matrices up to an arbitrary accuracy $\epsilon > 0$.

Proposition 1. *For any $n \in \mathbb{N}$, let $\mathcal{G}_n \subset \{0,1\}^{n\times n}$ denote the set of adjacency matrices of all n-node graphs. Given $K \in \mathbb{N}$ and $A \in \mathcal{G}_n$, the RRWP encoding is defined as:*

$$P = [\mathbf{I}, \mathbf{M}, \mathbf{M}^2, \dots, \mathbf{M}^{K-1}] \in \mathbb{R}^{n\times n\times K} \tag{30}$$

where $\mathbf{M} = \mathbf{D}^{-1}\mathbf{A}$ is the random walk transition matrix. Then, for any $\epsilon > 0$, there exists an MLP $MLP : \mathbb{R}^K \to \mathbb{R}$ acting independently across each dimension such that $MLP(P)$ approximates any of the following within ϵ error:

(a) $MLP(P)_{ij} \approx SPD_{K-1}(i,j)$, where $SPD_{K-1}(i,j)$ is the truncated shortest path distance.

(b) $MLP(P) \approx \sum_{k=0}^{K-1} \theta_k (\mathbf{D}^{-1}\mathbf{A})^k$, where $\theta_k \in \mathbb{R}$ are arbitrary coefficients.

(c) $MLP(P) \approx \theta_0\mathbf{I} + \theta_1\mathbf{A}$, where $\theta_0, \theta_1 \in \mathbb{R}$.

Proof. **(a) Approximation of Shortest Path Distance:** We construct an MLP such that $\mathrm{MLP}(P)_{ij} \approx \mathrm{SPD}_{K-1}(i,j)$, where $\mathrm{SPD}_{K-1}(i,j)$ is the shortest path length between nodes i and j if the path is at most $K-1$ hops, and takes the value K otherwise.

Define the RRWP matrix associated with adjacency matrix A as:

$$P(A) = [\mathbf{I}, \mathbf{D}^{-1}\mathbf{A}, \dots, (\mathbf{D}^{-1}\mathbf{A})^{K-1}]. \tag{31}$$

Let L be a lower bound on the smallest nonzero entry of $P(A)$ across all $A \in \mathcal{G}_n$, i.e.,

$$L = \min_{A\in\mathcal{G}_n} \min_{i,j,t: P(A)_{ijt}>0} P(A)_{ijt}. \tag{32}$$

Since the minimization is over a finite set of positive values, we have $L > 0$. Define the function $f_1 : \mathbb{R}^K \to \mathbb{R}^K$ such that:

$$f_1(x)_i = \begin{cases} 0, & x_i \le 0, \\ 1, & x_i \ge L. \end{cases} \tag{33}$$

Since for $t \in \{0, \dots, K-1\}$, $P_{ijt} \ge L$ if and only if node i is reachable from j in t hops, we obtain:

$$f_1(P_{i,j,:})_t = \begin{cases} 1, & \text{if } i \text{ can reach } j \text{ in } t \text{ hops}, \\ 0, & \text{otherwise}. \end{cases} \tag{34}$$

Next, define $f_2 : \mathbb{R}^K \to \mathbb{R}^K$ by:

$$f_2(x)_t = \max_{t' \le t} x_{t'}. \tag{35}$$

Thus, applying f_2 yields:

$$(f_2 \circ f_1(P_{i,j,:}))_t = \begin{cases} 1, & \text{if } \mathrm{SPD}(i,j) \le t, \\ 0, & \text{otherwise}. \end{cases} \tag{36}$$

Finally, define $f_3 : \mathbb{R}^K \to \mathbb{R}$ by:

$$f_3(x) = n - \sum_{t=1}^{K-1} x_t. \tag{37}$$

Applying this function gives:

$$(f_3 \circ f_2 \circ f_1)(P_{i,j,:}) = \begin{cases} \mathrm{SPD}(i,j), & \text{if } \mathrm{SPD}(i,j) \le K-1, \\ n, & \text{otherwise}. \end{cases} \tag{38}$$

Since $f_3 \circ f_2 \circ f_1$ is continuous, by the universal approximation theorem [?], an MLP can approximate it to arbitrary accuracy $\epsilon > 0$, completing the proof of (a).

(b) Approximation of Graph Propagation Matrices:

Given coefficients $\theta_0, \dots, \theta_{K-1} \in \mathbb{R}$, we construct an MLP such that:

$$\mathrm{MLP}(P) \approx \sum_{k=0}^{K-1} \theta_k (\mathbf{D}^{-1}\mathbf{A})^k. \tag{39}$$

Since this target function can be written as:

$$\sum_{k=0}^{K-1} \theta_k M^k = \sum_{k=0}^{K-1} \theta_k P_{:,:,k}, \tag{40}$$

we define the function $f_1 : \mathbb{R}^K \to \mathbb{R}^K$ that scales a vector $x \in \mathbb{R}^K$ elementwise by $\theta = [\theta_0, \dots, \theta_{K-1}]$:

$$f_1(x) = x \circ \theta. \tag{41}$$

Then, let $f_2 : \mathbb{R}^K \to \mathbb{R}$ sum over all elements:

$$f_2(x) = \sum_{k=0}^{K-1} x_k. \tag{42}$$

Thus, applying $f_2 \circ f_1$ to $P_{i,j}$ reconstructs the target function, which an MLP can approximate.

(c) Approximation of Weighted Adjacency Matrix:

Following similar steps as in (b), we approximate:

$$\text{MLP}(P) \approx \theta_0 \mathbf{I} + \theta_1 \mathbf{A}. \tag{43}$$

Using universal approximation arguments, an MLP can approximate this function, completing the proof. □

5.2.3 Flexible Attention Machenism

Build upon on RRWP, the Graph Inductive Bias Transformer (GRIT) introduces a novel **flexible attention mechanism** that jointly updates both **node representations** and **node-pair representations**. Unlike conventional Graph Transformers that primarily operate on node features, GRIT's attention module explicitly incorporates relational information between node pairs, ensuring richer and more structured updates. This section formally defines the flexible attention mechanism and its computational process.

Given a graph $G = (V, E)$ with n nodes, each node $i \in V$ is initialized with a feature vector $x_i \in \mathbb{R}^d$, and each node pair (i, j) is associated with an initial relational embedding $e_{i,j} \in \mathbb{R}^{d_e}$, which includes the learned RRWP positional encodings.

In each Transformer layer, GRIT simultaneously updates:

1. **Node representations** x_i based on its neighboring nodes.
2. **Node-pair representations** $e_{i,j}$, which refine the initial RRWP embeddings.

To achieve this, the attention scores are computed using a modified self-attention mechanism:

5.2.4 Initialization of Node and Node-Pair Representations

In each Transformer layer, we update both the **node representations** $x_i, \forall i \in \mathcal{V}$ and the **node-pair representations** $e_{i,j}, \forall i, j \in \mathcal{V}$. These representations are initialized using the input node features and the **Relative Random Walk Probabilities (RRWP)** positional encodings:

$$x_i = [x'_i \parallel P_{i,i}] \in \mathbb{R}^{d_h+K}, \quad e_{i,j} = [e'_{i,j} \parallel P_{i,j}] \in \mathbb{R}^{d_e+K} \tag{44}$$

where:

- $x'_i \in \mathbb{R}^{d_h}$ represents the observed node attributes.
- $e'_{i,j} \in \mathbb{R}^{d_e}$ represents the observed edge attributes.
- $P_{i,i}$ and $P_{i,j}$ are the RRWP encodings, capturing both absolute and relative positional relationships.
- If no observed edge exists between nodes i and j in the original graph, we set $e'_{i,j} = 0$.

5.2.5 Attention Computation with Absolute and Relative Representations

The proposed attention mechanism computes attention scores by incorporating both absolute and relative representations, ensuring a more expressive and flexible self-attention process:

$$\hat{e}_{i,j} = \sigma\left(\rho\left((W_Q x_i + W_K x_j) \odot W_E^w e_{i,j}\right) + W_E^b e_{i,j}\right) \in \mathbb{R}^{d'} \tag{45}$$

$$\alpha_{ij} = \text{Softmax}_{j \in \mathcal{V}}\left(W_A \hat{e}_{i,j}\right) \in \mathbb{R} \tag{46}$$

$$\hat{x}_i = \sum_{j \in \mathcal{V}} \alpha_{ij} \cdot (W_V x_j + W_E^v \hat{e}_{i,j}) \in \mathbb{R}^d \tag{47}$$

where:

- σ denotes a non-linear activation function (ReLU by default).
- $W_Q, W_K, W_E^w, W_E^b \in \mathbb{R}^{d' \times d}$, $W_A \in \mathbb{R}^{1 \times d'}$, and $W_V, W_E^v \in \mathbb{R}^{d \times d'}$ are learnable weight matrices.
- $\odot$ represents element-wise multiplication.
- $\rho(x) = (\text{ReLU}(x))^{1/2} - (\text{ReLU}(-x))^{1/2}$ is the **signed square-root** function, which stabilizes training by reducing the magnitude of large inputs.

Unlike standard self-attention mechanisms, this formulation allows for updating node-pair representations $e_{i,j}$, thereby enabling the Transformer to dynamically refine positional encodings. Specifically, this framework allows for an elementwise **MLP update to P**, which was shown to be beneficial in Proposition 3.1.

5.2.6 Extension to Multi-Head Attention

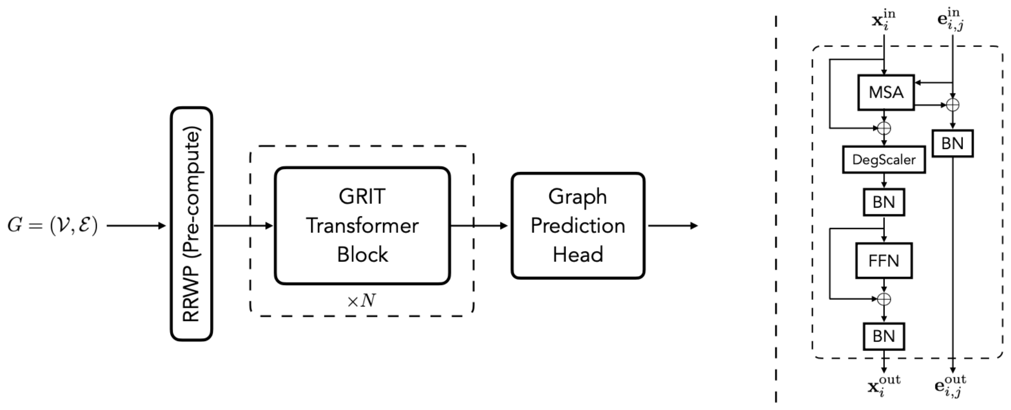
Similar to existing self-attention mechanisms, our proposed attention mechanism can be extended to **multi-head attention** with N_h heads. Each head h uses separate learnable parameters, producing head-specific representations $\hat{x}_i^h$ and $\hat{e}_{i,j}^h$, which are then aggregated as follows:

$$x_i^{\text{out}} = \sum_{h=1}^{N_h} W_O^h \hat{x}_i^h \in \mathbb{R}^d \tag{48}$$

$$e_{ij}^{\text{out}} = \sum_{h=1}^{N_h} W_E^h \hat{e}_{i,j}^h \in \mathbb{R}^d \tag{49}$$

where:

- $W_O^h, W_E^h \in \mathbb{R}^{d \times d'}$ are output weight matrices for each attention head h.
- The different heads enable the Transformer to capture diverse aspects of node and node-pair relationships.


$G = (\mathcal{V}, \mathcal{E})$
RRWP (Pre-compute)
GRIT Transformer Block
$\times N$
Graph Prediction Head
$\mathbf{x}_i^{\text{in}}$
$\mathbf{e}_{i,j}^{\text{in}}$
MSA
BN
DegScaler
BN
FFN
BN
$\mathbf{x}_i^{\text{out}}$
$\mathbf{e}_{i,j}^{\text{out}}$

(a) The Overall Architecture of GRIT Transformers

(b) The Architecture of a GRIT Transformer Block

Figure 7: 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.

5.3 Integration with Degree Information

To further enhance inductive bias, GRIT integrates degree information into the attention mechanism. Standard attention mechanisms are degree-invariant, which can reduce their expressivity. To address this, GRIT introduces an adaptive degree-scaler:

$$x_{\text{out},i} := x_{\text{out},i} \odot \theta_1 + (\log(1 + d_i) \cdot x_{\text{out},i} \odot \theta_2), \tag{50}$$

where d_i is the degree of node i, and $\theta_1, \theta_2 \in \mathbb{R}^d$ are learnable parameters.

Additionally, batch normalization (BN) is used instead of layer normalization to preserve degree information:

$$\mathrm{BN}(x_{\mathrm{out},i}) = \frac{x_{\mathrm{out},i} - \mu}{\sqrt{\sigma^2 + \epsilon}}, \tag{51}$$

where μ and σ^2 are computed across the batch.

By bridging the gap between message-passing and Transformer-based models, GRIT's flexible attention structure sets a new standard for graph representation learning.

6 Empirical Results ans Analysis

GraphGPS conducted a thorough empirical study and ablation analysis across all datasets, comparing various modules for message-passing based transformers. Therefore, in this report, we aim to complement current findings by further investigating spectral transformers such as GRIT. Specifically, we reproduce key baselines on a benchmark dataset to assess the impact of RRWP positional encoding and flexible attention mechanisms. By integrating RRWP positional encoding into a SAN transformer structure, we evaluate its contribution to model performance.

6.1 Dataset Introduction

ZINC is a widely used molecular dataset designed for benchmarking graph-based machine learning models, particularly in graph neural networks (GNNs) and graph Transformers. Derived from the ZINC database, which contains commercially available chemical compounds, the dataset focuses on molecular graph representations where nodes correspond to atoms and edges represent chemical bonds. A key task associated with ZINC is the prediction of molecular properties, such as penalized logP, a measure of drug-likeness that accounts for molecular solubility and synthesizability. The dataset consists of a subset of molecules with up to 38 heavy atoms, making it computationally feasible while maintaining structural diversity. Given its well-defined graph structure and relevance to real-world applications in drug discovery, ZINC has become a standard benchmark for evaluating the performance of graph-based learning models.

Table 1: Summary of the ZINC Dataset

Attribute	**Node-Level**	**Edge-Level**
Encoder Name	LapPE	TypeDictEdge
Number of Types	28	4
BatchNorm Applied	No	No
Dataset Format	PyG-ZINC	
Dataset Name	Subset	
Task Type	Graph Regression	
Transductive	No	

6.2 Training Process Overview

The table below lists the hyperparameter settings used for the experiment. The training was conducted for 400 epochs, which took approximately 400 minutes on a single NVIDIA A100 GPU. The training loss was monitored throughout the process as shown in figure 8 to ensure convergence.

Table 2: Key Training Hyperparameters

Training		**Optimizer and Scheduler**	
Hyperparameter	**Value**	**Hyperparameter**	**Value**
Batch Size	64	Learning Rate	0.001
Evaluation Period	1 epoch	Weight Decay	10^{-5}
Checkpoint Period	100 epochs	Scheduler	ConsineScheduler with Warmup
Graph Transformer (GT)		**Graph Neural Network (GNN)**	
Layers	10	Inner Dimension	56
Attention Heads	8	Activation Function	ReLU
Hidden Dimension	56	Aggregation Function	Mean

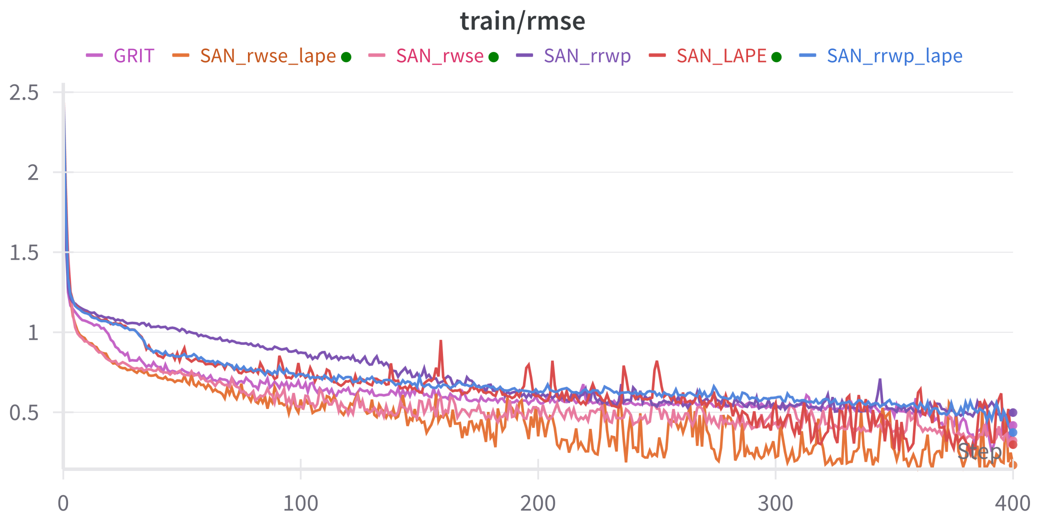
6.3 Evaluation Results

Table 3: Test Performance Comparison (RMSE, MSE, Params)

Model	**RMSE**	**MSE**	**Params**
GRIT	0.17824	0.02682	473921
SAN_rwse_lape	0.23558	0.05550	384509
SAN_rwse	0.24239	0.05875	384665
SAN_rrwp	0.47630	0.22686	384665
SAN_LAPE	0.44189	0.19527	454753
SAN_rrwp_lape	0.42164	0.17778	454753

We tracked key metrics such as validation loss and accuracy during the training process, as shown in Figure 8. Overall, GRIT and SAN with RWSE exhibit the best performance. While SAN with RWSE achieves a lower training loss, GRIT demonstrates greater potential for generalization given additional training epochs. The final evaluation metrics are presented in Table 3, highlighting several key insights:

- For SAN, the default Laplacian Positional Encoding (LaPE) is not necessarily the most optimal setting. Our results indicate that random-walk-based positional encoding methods can also yield strong performance, providing a viable alternative to LaPE.



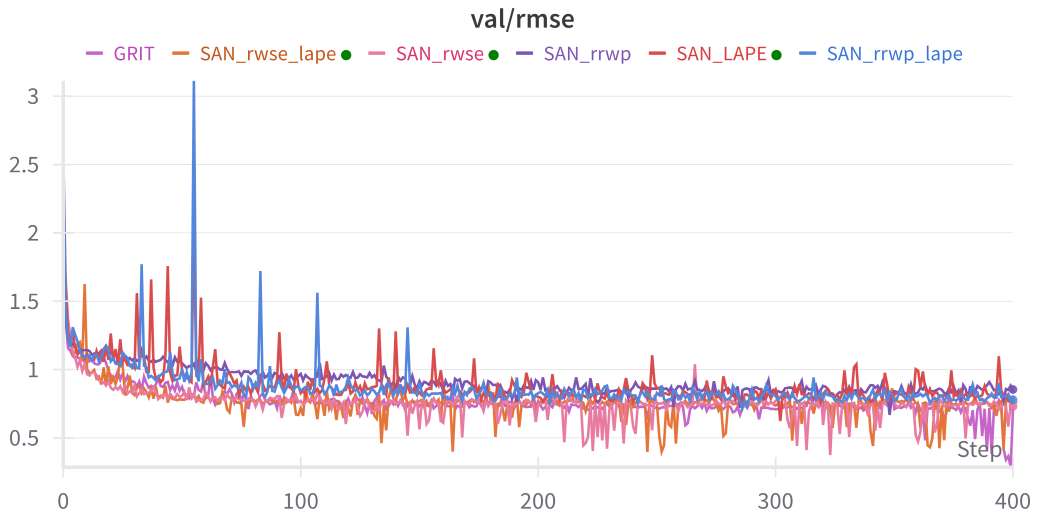


Figure 8: Tracked training and validation loss for GRIT and SAN with different positional encoding strategies on the ZINC dataset.

- The application of RRWP on top of LaPE offers some improvements; however, RRWP alone leads to the worst results. This suggests that absolute positional embeddings play a crucial role in SAN, aiding the model in effectively capturing structural information.
- The combination of RRWP with GRIT's flexible attention mechanism yields the best results, indicating a strong coupling between these components. This synergy suggests that RRWP benefits from the adaptive nature of GRIT's attention mechanism, enhancing its ability to model complex relationships in graph data.

These findings underscore the importance of selecting appropriate positional encodings based on the specific characteristics of the dataset and model architecture. Future work could further investigate optimal configurations of absolute and relative positional encodings, exploring their interactions with different graph neural network architectures and training regimes. This version enhances clarity, improves structure, and presents the findings in a more formal,