LEARNING EFFICIENT POSITIONAL ENCODINGS WITH GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Positional encodings (PEs) are essential for effective graph representation learning because they provide position awareness in inherently position-agnostic transformer architectures and increase the expressive capacity of Graph Neural Networks (GNNs). However, designing powerful and efficient PEs for graphs poses significant challenges due to the absence of canonical node ordering and the scale of the graph. Here, we investigate PEs for graphs based on four key criteria: stability, expressive power, scalability, and genericness. We find that existing eigenvector-based PE methods often fall short of jointly satisfying these criteria. To address this gap, we introduce PEARL, a novel framework of learnable PEs for graphs. Our primary insight is that message-passing GNNs function as nonlinear mappings of eigenvectors, enabling the design of GNN architectures for generating powerful and efficient PEs. A crucial challenge lies in initializing node attributes in a manner that is both expressive and permutation equivariant. We tackle this by initializing GNNs with random node inputs or standard basis vectors, thereby unlocking the expressive power of message-passing operations, while employing statistical pooling functions to maintain permutation equivariance. Our analysis demonstrates that PEARL approximates equivariant functions of eigenvectors with linear complexity, while rigorously establishing its stability and high expressive power. Experimental evaluations show that PEARL outperforms lightweight versions of eigenvector-based PEs and achieves comparable performance to full eigenvector-based PEs, but with one or two orders of magnitude lower complexity.

1 INTRODUCTION

Positional encodings (PEs) are a fundamental component of graph representation learning and play a key role in the design of effective Graph Transformers (Dwivedi & Bresson, 2021; Rampášek et al., 2022) and Graph Neural Networks (GNNs) (Kipf & Welling, 2016; Hu et al., 2020). Transformer architectures (Vaswani, 2017) are inherently agnostic to structure and node identities, and PEs provide a powerful mechanism to incorporate positional and structural information. On the other hand, message-passing GNNs often struggle with low expressiveness, especially when node attributes exhibit the same symmetries as the graph structure (Xu et al., 2019; Morris et al., 2019; Kanatsoulis & Ribeiro, 2024). By integrating structural and positional information, PEs enhance GNNs' capacity to capture patterns that would otherwise be difficult to learn and generalize.

Several graph PE methods have been proposed in the literature, which can broadly be categorized into two main types: absolute PEs and relative PEs. Absolute PEs assign an embedding to each node in the graph, reflecting the node's role within the graph structure. Common approaches include Laplacian eigenvectors (Dwivedi & Bresson, 2021), substructure encodings (Tahmasebi et al., 2020; You et al., 2021; Bouritsas et al., 2022), random walk (RW) encodings (Rampášek et al., 2022), and eigenvector-based methods (Kreuzer et al., 2021; Lim et al.; Huang et al.). Relative PEs, on the other hand, assign representations to pairs of nodes and typically utilize measures such as shortest-path and resistance distances (Ying et al., 2021; Zhang et al., 2023), as well as RW matrices (Ma et al., 2023; Geisler et al., 2023). A thorough comparison between absolute and relative PEs can be found in (Black et al., 2024).

In this paper, we study absolute PEs for graphs based on four key criteria: expressive power, scalability, stability under perturbations, and generality. We find that PEs based on eigenvectors of

graph Laplacian or other graph operators often struggle to satisfy all these criteria simultaneously. To better understand this, we divide eigenvector-based approaches into two categories: those that compute the full set of eigenvectors and those that only consider the K largest. Full eigenvector approaches offer high expressive power but come with a computational complexity of $\mathcal{O}(N^3)$ and memory complexity of $\mathcal{O}(N^2)$, which is prohibitive for even medium-sized graphs. The full set of eigenvectors can also be used to learn spectral graph filters (Huang et al.), which result in stable PEs. Note that stability is particularly crucial for out-of-distribution generalization.

However, when only a subset of eigenvectors is computed, several limitations arise. First, this introduces an inductive bias, as different graphs encode different information across eigenvalues, especially when they differ in size. Second, the expressive power and stability are reduced, becoming dependent on the eigengap between the selected eigenvalues. These methods also face challenges in terms of stability and generalization when applied to different or unseen graph structures. Consequently, this approach often leads to significantly poorer performance. Substructure-based encodings face similar challenges: while generally stable, they also introduce inductive bias and highly expressive versions require combinatorial complexity. The aforementioned challenges raise a critical research question:

Question: Can we learn generic PEs that are simultaneously expressive, stable, and scalable?

In this work, we provide an affirmative answer by proposing PEARL, a powerful and efficient framework for learnable PEs, entirely generated via message-passing GNNs. We begin by showing that message-passing GNNs can be understood as nonlinear mappings of eigenvectors of the graph Laplacian or other graph shift operators. This insight enables the computation of eigenvector-based PEs efficiently with linear or quadratic complexity, leveraging message-passing operations. A central challenge in developing effective PEs with GNNs lies in initializing node attributes to ensure both expressiveness and permutation equivariance. We address this by initializing each node with a set of M random samples, effectively breaking the symmetries between the graph structure and node attributes. Each sample is processed independently by a GNN, and to guarantee permutation equivariance, we design pooling functions based on statistics. Our analysis demonstrates that PEARL surpasses the expressiveness of the Weisfeiler-Leman (WL) test (Weisfeiler & Leman, 1968), and is capable of counting key substructures at the node level.

PEARL is provably stable, inheriting the stability guarantees of GNNs (Gama et al., 2020), which are independent of the eigenvalue gap. Moreover, we analyze the sample complexity of PEARL and show that the number of samples required for effective encoding is independent of graph size. This enables the generation of powerful eigenvector-based PEs for large graphs with linear complexity. For smaller graphs, where the number of samples is comparable to the graph size, we propose an alternative model that initializes node attributes with basis vectors. This approach approximates the PEs in (Huang et al.) with significantly lower computational and memory complexity. We evaluate the proposed PEARL on graph classification and regression tasks on molecular graphs and social network datasets, and compare it against eigen-based and structure-based absolute PEs. The results demonstrate that PEARL consistently outperforms structure-based PEs and lightweight variants of eigenvector-based PEs, achieving up to a 6% improvement on graph classification tasks. In comparison to full eigenvector-based PEs, which have a computational complexity of $\mathcal{O}(N^3)$, PEARL delivers comparable performance with significantly reduced complexity, scaling at $\mathcal{O}(N)$ or $\mathcal{O}(N^2)$.

2 PRELIMINARIES

A graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$, is represented by a set of vertices $\mathcal{V} = \{1, \dots, N\}$, a set of edges $\mathcal{E} = \{(v, u)\}$, and a graph shift operator (GSO) $\boldsymbol{S} \in \mathbb{R}^{N \times N}$. The GSO is typically sparse, with common choices including the adjacency matrix, the Laplacian matrix, their normalized variants, or the RW transition matrix. The nodes (vertices) in the graph are often associated with node signals $\boldsymbol{x}_v \in \mathbb{R}^d$, each with d features, while edges can carry edge attributes $\boldsymbol{x}_{(u,v)} \in \mathbb{R}^{d_e}$ with d_e features.

An important operation in graph theory and network science is the spectral decomposition of the graph and refers to the eigenvalue decomposition to the GSO, $\boldsymbol{S} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T$. Matrix $\boldsymbol{V} = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_n]$ is the orthonormal matrix of eigenvectors, and $\boldsymbol{\Lambda}$ is the diagonal matrix of eigenvalues $\{\lambda_n\}_{n=1}^N$. When $\boldsymbol{S}$ represents the Laplacian matrix, $\boldsymbol{V}$ are the Laplacian eigenvectors that are commonly used as node features or positional encodings for GNN architectures.

In this paper, we study standard message-passing GNNs, defined by the following recursive formula:

$$\boldsymbol{x}_v^{(l)} = g^{(l-1)}\left(\boldsymbol{x}_v^{(l-1)}, f^{(l-1)}\left(\left\{\boldsymbol{x}_u^{(l-1)} : u \in \mathcal{N}(v)\right\}\right)\right). \tag{1}$$

Here, $\mathcal{N}(v)$ represents the neighborhood of vertex v, meaning that $u \in \mathcal{N}(v)$ if and only if $(u, v) \in \mathcal{E}$. The function $f^{(l)}$ aggregates information from the multiset of signals coming from neighboring vertices, while $g^{(l)}$ combines the signal of each vertex with the aggregated information from its neighbors. Common choices for $f^{(l)}$ and $g^{(l)}$ include the single- and multi-layer perceptron (MLP), the linear function, and the summation function.

3 Our Work: Learnable, Efficient, and Powerful PEs with GNNs

3.1 GNNs are nonlinear functions of GSO eigenvectors

Our first observation is that message-passing GNNs are nonlinear functions of eigenvectors. To see this, let $f^{(l)}$ be one of the following aggregation functions:

$$\sum_{u\in\mathcal{N}(v)} \boldsymbol{x}_u,\ d_v \cdot \boldsymbol{x}_v - \sum_{u\in\mathcal{N}(v)} \boldsymbol{x}_u,\ \sum_{u\in\mathcal{N}(v)} \frac{\boldsymbol{x}_u}{\sqrt{d_v d_u}},\ \boldsymbol{x}_u - \sum_{u\in\mathcal{N}(v)} \frac{\boldsymbol{x}_u}{\sqrt{d_v d_u}},\ \sum_{u\in\mathcal{N}(v)} \frac{\boldsymbol{x}_u}{d_u} \tag{2}$$

where d_v is the degree of node v. Then Eq. (1) can be written as $\boldsymbol{X}^{(l)} = g^{(l-1)}\left(\boldsymbol{X}^{(l-1)}, \boldsymbol{S}\boldsymbol{X}^{(l-1)}\right)$, where $\boldsymbol{S}$ represents the adjacency matrix, the Laplacian matrix, the normalized adjacency, the normalized Laplacian, and the RW matrix, for the five choices of $f^{(l)}$ in 2 respectively, and $\boldsymbol{X}^{(l)} \in \mathbb{R}^{N\times F_l}$ represents the signals of all vertices at layer l. Now let $g^{(l)}$ be an equivariant MLP operating on each node independently. Note that the MLP is a common choice for function $g^{(l)}$ for the majority of effective GNN architectures due to its expressiveness properties. Then Eq. (1) can be cast as:

$$\boldsymbol{X}^{(l)} = \sigma\left(\sum_{k=0}^{K-1} \boldsymbol{S}^k \boldsymbol{X}^{(l-1)} \boldsymbol{H}_k^{(l)}\right), \tag{3}$$

where $K = 2$, $\boldsymbol{H}_k \in \mathbb{R}^{F_{l-1}\times F_l}$ are the trainable parameters, and σ is a point-wise nonlinear activation function. Note that Eq. (3) defines a single-layer graph perceptron, but it can be easily generalized to a multi-layer graph perceptron by letting σ represent an equivariant MLP acting on the node signals. Additionally, while we set $K = 2$ here, higher values of K can be considered for more generalized GNN layers. It is worth emphasizing that $\boldsymbol{S}^k$ is never explicitly instantiated; instead, $\boldsymbol{S}^k\boldsymbol{X}^{(l-1)}$ is computed using recursive message-passing operations, as outlined in Eq. (2).

Proposition 3.1 (GNNs are nonlinear functions of eigenvectors) *A GNN defined in Eq. (1) with $f^{(l)}$ being one of the functions in Eq. (2) and $g^{(l)}$ being a multi-layer perceptron, operates as a nonlinear function of the GSO eigenvectors i.e., $\boldsymbol{x}_v^{(l)} = \texttt{MLP}\left(\boldsymbol{v}^{(v)}\right)$, $\boldsymbol{v}^{(v)} = \boldsymbol{V}[v,:]^T$. The trainable parameters of the first $\texttt{MLP}$ layer are not independent but depend on the eigenvalues $\{\lambda_n\}_{n=1}^N$ and eigenvectors $\{\boldsymbol{v}_n\}_{n=1}^N$ of the GSO, as well as the node features $\boldsymbol{X}$ of the graph:*

$$\boldsymbol{x}_v^{(l)} = \texttt{MLP}\left(\boldsymbol{v}^{(v)}\right) = \texttt{MLP}^{(-1)}\left(\sigma\left(\boldsymbol{W}^T\boldsymbol{v}^{(v)}\right)\right) \tag{4}$$

$$\boldsymbol{W}[n,f] = \sum_{i=1}^{F_{l-1}}\sum_{k=0}^{K-1} \lambda_n^k \boldsymbol{H}_k^{(l)}[i,f]\langle\boldsymbol{\alpha}_n, \boldsymbol{X}^{(l-1)}[:,i]\rangle, \tag{5}$$

where $\boldsymbol{\alpha}_n = \boldsymbol{v}_n$ when the GSO is symmetric and $\boldsymbol{\alpha}_n = \boldsymbol{V}^{-1}[n,:]$ when it is not. $\texttt{MLP}^{(-1)}$ denotes all the layers of the $\texttt{MLP}$ except the first layer.

The proof is provided in Appendix B. According to Proposition 3.1, the update for node v, defined by the function $g \circ f : \left(\mathcal{G}, \mathbb{R}^{F_{l-1}}\right) \to \mathbb{R}^{F_l}$, can be interpreted as a nonlinear mapping ($\texttt{MLP}$) applied to $\boldsymbol{V}[v,:]$, but the weights of the first layer of this mapping are also mappings, i.e., $\mathbf{W}[n,f] : \left(\mathcal{G}, \mathbb{R}^{F_{l-1}}\right) \to \mathbb{R}^1$. The degrees of freedom in the first layer of $\texttt{MLP}$ are KF_lF_{l-1} (as described in Eq. (5)), rather than F_lN, which would be the case for independent weights $\boldsymbol{W}$. Furthermore, the dot product $\langle\boldsymbol{\alpha}_n, \boldsymbol{X}^{(l-1)}[:,i]\rangle$ depends on the eigenvectors and, for the update of node v, it only involves the components $\boldsymbol{X}^{(l-1)}[u,i],\ u \in \mathcal{N}_v$. Proposition 3.1 is applicable to most message-passing GNN models, including, but not limited to, Graph Convolutional Networks (GCNs) (Kipf & Welling, 2016), Graph Isomorphism Networks (GINs) (Xu et al., 2019), and GraphSAGE (Hamilton et al., 2017).

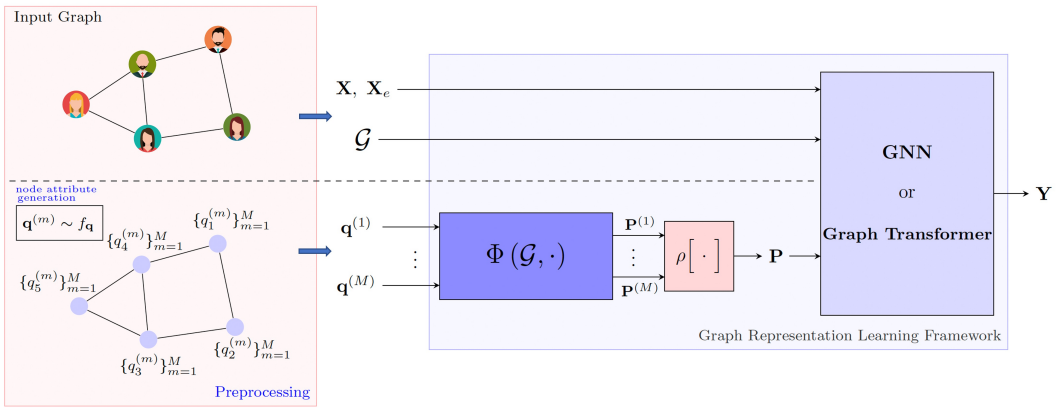


Figure 1: PEARL framework: The input graph undergoes anonymization by removing its node and edge attributes. For each node, a set of M random or basis attributes is generated. Each sample is then independently processed by a message-passing GNN, and a pooling function ρ is applied to produce equivariant PEs. The graph structure, together with the generated PEs and any node or graph attributes, is subsequently processed using either a GNN or a Graph Transformer.

3.2 PEARL: Expressive and Equivariant Positional Encoding Networks

Following the derivation of Proposition 3.1, a critical question arises: what is the optimal choice of node attributes that allow a GNN to compute expressive and equivariant functions of the eigenvectors? Equivariant structural features augment GNNs with valuable information, but they come at the cost of increased computational complexity and inductive bias. Moreover, these features share the same symmetries as the graph structure, limiting the expressiveness of message-passing GNNs. Alternatively, unique identifiers, such as random node features, can break the structural symmetries and improve expressiveness but at the expense of permutation equivariance, which limits the model's generalization capability. To address this trade-off between equivariance and expressiveness, we propose to momentarily break the structural symmetries by initializing each node with a set of M unique identifiers, while maintaining permutation equivariance in the model output via the use of statistical pooling functions. The proposed PE framework (PEARL) is illustrated in Fig. 1 and, as we see next, ensures both high expressiveness and strong generalization.

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes. For each node $v \in \mathcal{V}$ in $\mathcal{G}$, we design a set of M $1-$dimensional node signals $\left\{q_v^{(1)}, q_v^{(2)}, \ldots, q_v^{(M)}\right\}$, where each $q_v^{(m)}$ operates as a unique identifier. Graph $\mathcal{G}$ is now associated with a set of M independent initial node attributes represented as $\left\{\boldsymbol{q}^{(m)}\right\}_{m=1}^M$, $\boldsymbol{q}^{(m)} \in \mathbb{R}^N$. Each pair of $\left\{\mathcal{G}, \boldsymbol{q}^{(m)}\right\}$ is independently processed via a GNN $\Phi(\cdot)$, which is described by Eq. (1) or (3), to produce a set of M independent outputs:

$$\boldsymbol{P}^{(m)} = \Phi\left(\mathcal{G}, \boldsymbol{q}^{(m)}\right) \in \mathbb{R}^{N \times d_p}, \quad m = 1, \ldots, M \tag{6}$$

Since $\left\{\boldsymbol{q}^{(m)}\right\}_{m=1}^M$ operate as unique identifiers, they break the structural symmetries and unlock the expressive power of message-passing operations. However, each $\boldsymbol{P}^{(m)}$ is not permutation equivariant, thus not generalizable. To address this, we leverage the independence among $\{\boldsymbol{P}^{(m)}\}_{m=1}^M$ and design an equivariant pooling function ρ to generate the final PE for each node:

$$\boldsymbol{P} = \rho\left[\Phi\left(\mathcal{G}, \boldsymbol{q}^{(1)}\right), \ldots, \Phi\left(\mathcal{G}, \boldsymbol{q}^{(M)}\right)\right] \in \mathbb{R}^{N \times d_p} \tag{7}$$

The PEARL framework can universally approximate any continuous basis invariant function.

Theorem 3.1 (Basis Universality) *Let $\mathcal{G}$ be a graph with GSO $\boldsymbol{S} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T$, and f be a continuous function such that $f(\boldsymbol{V}) = f(\boldsymbol{V}\boldsymbol{Q})$, $\boldsymbol{Q} \in \mathcal{O}(diag(\boldsymbol{\Lambda}))$, for any eigenvalues $\boldsymbol{\Lambda}$. Then there exist GNN Φ and a continuous pooling function ρ, such that $f(\boldsymbol{V}) = \rho\left[\Phi\left(\mathcal{G}, \boldsymbol{q}^{(1)}\right), \ldots, \Phi\left(\mathcal{G}, \boldsymbol{q}^{(M)}\right)\right]$.*

The proof can be found in Appendix C. In the following sections, we explore options for the initial node attributes $\left\{\boldsymbol{q}^{(m)}\right\}_{m=1}^{M}$ and pooling functions ρ. A key aspect of PEARL is designing M independent initial attributes for each node, which enables permutation equivariance at the model's output through the use of pooling functions. This stands in contrast to classical methods, which typically assign a single unique identifier per node.

4 Our work: Random Positional Encoding Network (R-PEARL)

Next, we present our Random PE Network (R-PEARL). In R-PEARL we define node attributes $\left\{\boldsymbol{q}^{(m)}\right\}_{m=1}^{M}$ by sampling them randomly from a probability distribution. Specifically, let $\boldsymbol{q} = [q_{v_1}, q_{v_2}, \ldots, q_{v_N}]^T$, where $v_n \in \mathcal{V}$, be a random vector with joint distribution $f_{\boldsymbol{q}}(t_1, \ldots, t_N)$. The set $\left\{\boldsymbol{q}^{(m)}\right\}_{m=1}^{M}$ consists of M independent N-dimensional realizations of $\boldsymbol{q}$, drawn from $f_{\boldsymbol{q}}$. In our experiments and analysis, $\boldsymbol{q}$ is either a set of independent and identically distributed (i.i.d.) Gaussian random variables or a set of i.i.d. random variables with $\mathbb{E}[q_i] = 0$ and $\mathbb{E}[q_i^p] = 1$, where $p \geq 2$.

When these samples are processed by a GNN Φ, the result is M $(N \times d_p)$-dimensional samples of the random matrix output $\Phi(\mathcal{G}, \boldsymbol{q})$. To practically preserve permutation equivariance, we note that the distribution of $\Phi(\mathcal{G}, \boldsymbol{q})$ is itself permutation equivariant, as are any statistics derived from it. Therefore, the function ρ can be any empirical statistic computed from the samples $\left\{\Phi(\mathcal{G}, \boldsymbol{q}^{(m)})\right\}_{m=1}^{M}$, each capturing different characteristics. For instance, ρ could represent any statistical moment, such as the mean or variance, or other measures as the empirical mode or median. In this paper, we choose ρ to be the empirical mean due to its favorable convergence and stability properties, as well as its simplicity in implementation and low computational and memory complexity:

$$\boldsymbol{P} = \hat{\mathbb{E}}\left[\Phi\left(\mathcal{G}, \boldsymbol{q}^{(1)}\right), \ldots, \Phi\left(\mathcal{G}, \boldsymbol{q}^{(M)}\right)\right] = \frac{1}{M}\sum_{m=1}^{M}\Phi\left(\mathcal{G}, \boldsymbol{q}^{(m)}\right) = \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{P}^{(m)} \tag{8}$$

In Appendix D, we explicitly analyze the equivariant functions learned by Eq. (8). We derive nonlinear expressions both in the graph domain, using vertex and edge information, and in the frequency domain, using the eigenvectors and eigenvalues of the GSOs. The key to this nonlinear analysis involves studying the pointwise nonlinearities through their Taylor series expansion.

4.1 Sample Complexity

In this section, we analyze the number of samples required to such that $\frac{1}{M}\sum_{m=1}^{M}\boldsymbol{P}^{(m)}$ approximates $\mathbb{E}[\Phi(\mathcal{G}, \boldsymbol{q})]$ with negligible error. To that end, we make the following two assumptions that will be later used for the stability analysis as well.

Assumption 4.1 *The pointwise nonlinearity σ is Lipschitz continuous with Lipschitz constant C_σ.*

This is a common assumption in deep learning and is satisfied by the widely used nonlinearities. In most cases, such as the Rectified Linear Unit (ReLU), hyperbolic tangent, and sigmoid, it holds that $C_\sigma = 1$. Before introducing the second assumption, we first need to examine Eq. (3) more closely. Notice that its linear component involves $F_L \cdot F_{l-1}$ graph filters of the form $\sum_{k=0}^{K-1} h_k \boldsymbol{S}^k$, which is also explicitely shown in Appendix E, Eq. (48).

Assumption 4.2 *The linear operators $\boldsymbol{H}(\boldsymbol{S}) = \sum_{k=0}^{K-1} h_k \boldsymbol{S}^k$ involved in the projection of Eq. (3) are bounded, i.e., $\|\boldsymbol{H}(\boldsymbol{S})\| \leq \beta$.*

This is another common assumption in deep learning, where the value of β varies depending on the architecture and task. We can now present Theorem 4.3, which characterizes the number of samples M needed for our approach to converge to the true $\mathbb{E}[\Phi(\mathcal{G}, \boldsymbol{q})]$.

Theorem 4.3 (Sample Complexity) *Let $\boldsymbol{P}$ denote the output of the architecture described in Eq. (8), for a graph $\mathcal{G}$ with i.i.d. initial node attributes with unit variance. Also let Φ be an $L-$layer GNN described by Eq. (3), with F hidden dimensions at each layer. If $C_\sigma = 1$ and $\beta = 1/F$, the number*

of samples M required such that:

$$\left| \frac{1}{M} \sum_{m=1}^{M} \Phi\left(\mathcal{G}, \boldsymbol{q}^{(m)}\right) - \mathbb{E}\left[\Phi\left(\mathcal{G}, \boldsymbol{q}\right)\right] \right| < \epsilon, \text{ with probability at least } 1-\delta, \tag{9}$$

satisfies:

$$M \leq \frac{1}{\delta \cdot \epsilon^2}. \tag{10}$$

It is worth noting that the above bound is independent of the size of the graphs, which suggests that our proposed PE framework is well-suited for large-scale graphs. In practice, we have observed that $10 \sim 100$ samples are typically sufficient.

4.2 Expressive Power

In this section, we establish the expressive power of our proposed R-PEARL.

Corollary 4.4 (Structure Counting) *Let $\boldsymbol{q} = [q_1, \ldots, q_N]$ be a set of N i.i.d. random variables such that $\mathbb{E}[q_i] = 0$ and $\mathbb{E}[q_i^p] = 1$ for $p \geq 2$. Then, there exists a parametrization Φ, defined by Eq. (3), such that $\mathbb{E}\left[\Phi\left(\cdot, \boldsymbol{q}\right)\right]$ counts the number of 3-, 4-, 5-, 6-, and 7-node cycles in which each node participates, for any given graph.*

Corollary 4.4 not only highlights the expressive power of R-PEARL framework but also provides valuable insights into its generalization ability. Essentially, R-PEARL framework can learn not just the number of cycles each node in a given graph participates in, but also a counting function that generalizes this capability to any node in any graph. The proof can be found in Appendix F, and is based on the results in (Kanatsoulis & Ribeiro). In Corollary 4.5 we characterize the expressive power of message-passing GNNs with our proposed PEs with respect to the folklore-Weisfeiler-Leman (FWL) test (Cai et al., 1992; Morris et al., 2019; Huang & Villar, 2021).

Corollary 4.5 (Expressive Power) *A GNN defined in Eq. (1), with PEs produced by Eq. (8) is strictly more powerful than the 1-FWL test, when f, g are injective functions.*

The proof of Corollary 4.5 is a consequence of Corollary 4.4 and the analysis in (Xu et al., 2019). Note that the previous results can be improved (e.g., count cycles and cliques of higher order, go beyond 2-FWL test) when the samples $\left\{\boldsymbol{q}^{(m)}\right\}_{m=1}^{M}$ are drawn from a structurally aware distribution, but this will increase the number of computations and is outside of the scope of this paper.

4.3 Stability

The proposed PEs are purely generated by GNN architectures and as a result they inherit favorable stability properties of GNNs. Any stability results for GNNs hold for R-PEARL as well. For instance, let $\tilde{\mathcal{G}}$ be a perturbed version of $\mathcal{G}$ such that $\tilde{\boldsymbol{S}} = \boldsymbol{S} + \boldsymbol{E}$. We can use the stability results in (Gama et al., 2020) and derive the following proposition.

Corollary 4.6 (Stability) *Let $\tilde{\mathcal{G}}$ be a perturbed version of $\mathcal{G}$ such that $\tilde{\boldsymbol{S}} = \boldsymbol{S} + \boldsymbol{E}$ with $\|\boldsymbol{E}\| \leq \varepsilon$. Let Φ be an $L-$layer GNN described by Eq. (3), where each layer consists of F^2 Lipschitz continuous filters [cf. Eq. (G.2)] with constant C. Under assumptions 4.1 and 4.2 with $C_\sigma = 1$ and $\beta = 1/F$, it holds that:*

$$\left\| \frac{1}{M} \sum_{m=1}^{M} \Phi\left(\mathcal{G}, \cdot\right)[:, f] - \frac{1}{M} \sum_{m=1}^{M} \Phi\left(\tilde{\mathcal{G}}, \cdot\right)[:, f] \right\|_{\mathcal{P}} \leq \left(1 + 8\sqrt{N}\right) L\varepsilon + \mathcal{O}(\varepsilon^2) \tag{11}$$

where $\|\cdot\|_{\mathcal{P}}$ is the distance modulo permutation [cf. G.1], and M is the number of samples.

We can further normalize the proposed PEs by $\sqrt{N} \cdot L$ to improve the stability bound. Notably, our result remains independent of the eigengap δ_λ, which is the difference between consecutive eigenvalues of the GSO. However, this independence does not hold for the stability of eigenvectors. According to the Davis-Kahan Theorem (Davis & Kahan, 1970), even a small perturbation in the

graph can lead to arbitrarily large differences between the eigenvector encodings of the original and perturbed GSOs. This limitation also applies to the eigenvector-based PEs in (Lim et al.). The stability bound of the PEs in (Huang et al.) is inversely proportional to the eigengap δ_λ between the d-th and $(d+1)$-th smallest eigenvalues when using the first d eigenvectors. This dependence is mitigated if all eigenvectors are computed, but doing so requires $\mathcal{O}(N^3)$ complexity, which is impractical for large graphs. Further details on stability results, are provided in Appendix G.

4.4 Computational complexity

To implement R-PEARL, we process each initial random attribute independently using a message-passing GNN. Consequently, the computational complexity of the feed-forward pass is equivalent to that of a message-passing GNN multiplied by the number of samples, i.e., $\Theta\left(MNF^2 + M|\mathcal{E}|F\right)$, where F represents the hidden dimension of each GNN layer. The memory complexity of a serial implementation is $\Theta\left(NF\right)$, while for a parallel implementation, it becomes $\Theta\left(MNF\right)$.

5 Our work: Basis Positional Encoding Networks (B-PEARL)

The previous approach R-PEARL is particularly advantageous for large graphs, where the number of samples is much smaller than the number of nodes and edges, making the computational and memory complexity approximately linear. However, for smaller-scale graphs, such as molecular graphs, the computational complexity becomes quadratic. In these cases, we propose using standard basis vectors $\{\boldsymbol{e}_m\}_{m=1}^N$ as the initial node attributes, where $\boldsymbol{e}_m[m] = 1$ and $\boldsymbol{e}_m[i \neq m] = 0$, thus setting $M = N$. Similar to the previous approach, when these samples are processed by a GNN Φ, the result is N $(N \times d_p)$-dimensional outputs. To maintain permutation equivariance, any equivariant function ρ can be applied, but in this paper, we choose the summation pooling for ρ. Overall, B-PEARL is cast as:

$$\boldsymbol{P} = \rho\left[\Phi\left(\mathcal{G}, \boldsymbol{e}_1\right), \ldots, \Phi\left(\mathcal{G}, \boldsymbol{e}_N\right)\right] = \sum_{m=1}^{N} \Phi\left(\mathcal{G}, \boldsymbol{e}_m\right) = \sum_{m=1}^{N} \boldsymbol{P}^{(m)} \tag{12}$$

5.1 Relation to Eigenvector based encodings

The proposed B-PEARL framework is highly related to the stable and expressive positional encodings (SPE) proposed in (Huang et al.). In particular SPE is defined as $\text{SPE}\left(\boldsymbol{V}, \boldsymbol{\Lambda}\right) = \rho\left(\left[\boldsymbol{V}\text{diag}\left(\alpha_1\left(\boldsymbol{\Lambda}\right)\right)\boldsymbol{V}^T, \ldots, \boldsymbol{V}\text{diag}\left(\alpha_F\left(\boldsymbol{\Lambda}\right)\right)\boldsymbol{V}^T\right]\right)$, where $\{\alpha_i\}_{i=1}^F$ is a set of continuous functions and ρ is an equivariant function. The suggested SPE implementation is $\text{SPE}\left(\boldsymbol{V}, \boldsymbol{\Lambda}\right) = \sum_{n=1}^{N} \Phi\left(\left[\boldsymbol{V}\text{diag}\left(\alpha_1\left(\boldsymbol{\Lambda}\right)\right)\boldsymbol{V}[n], \ldots, \boldsymbol{V}\text{diag}\left(\alpha_M\left(\boldsymbol{\Lambda}\right)\right)\boldsymbol{V}[n]\right]\right)$, where Φ is a GNN, and $\{\alpha_i\}_{i=1}^F$ are MLPs. The computational complexity is cubic with respect to the number of nodes and the memory complexity is quadratic.

Remark 5.1 *When $\{\alpha_i\}_{i=1}^F$ in SPE $\left(\boldsymbol{V}, \boldsymbol{\Lambda}\right)$ are pointwise analytic functions, i.e., the pointwise nonlinearities are differentiable, the SPE architecture is equivalent to the proposed B-*PEARL *architecture in Eq. (12). The proof can be found in Appendix H.*

5.2 Computational complexity

To implement B-PEARL, we process each initial basis encoding independently using a message-passing GNN. As a result, the computational complexity is $\Theta\left(N^2F^2 + N|\mathcal{E}|F\right)$, where F represents the hidden dimension of each GNN layer. The memory complexity for a serial implementation is $\Theta\left(NF\right)$, while for a parallel implementation, it increases to $\Theta\left(N^2F\right)$.

6 Experiments

In this section, we assess the performance of PEARL on graph classification and regression tasks. All experiments were conducted on a Linux server with NVIDIA A100 GPU. Code can be found in this repository[1].

[1] https://github.com/codelakepapers/RPE-Framework

Table 1: Graph classification accuracy on REDDIT-B and REDDIT-M (OOM stands for out-of-memory). R-PEARL outperforms all light-weight baselines by at least 2.5% in REDDIT-B and 3.5 % in REDDIT-M. It also achieves better performance compared to SPE in RREDDIT-M and comparable in in REDDIT-B, but with much lower complexity.

Method	Computational Complexity	Memory Complexity	REDDIT-B	REDDIT-M
GCN	$\mathcal{O}(N)$	$\mathcal{O}(N)$	50.0 ± 0.0	20.0 ± 0.0
GIN	$\mathcal{O}(N)$	$\mathcal{O}(N)$	91.8 ± 1.0	56.9 ± 2.0
GIN + rand id	$\mathcal{O}(N)$	$\mathcal{O}(N)$	91.8 ± 1.6	57.0 ± 2.1
GSN with cliques	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$	91.1 ± 1.8	56.2 ± 1.8
SignNet-8S	$\mathcal{O}(N^3)$	$\mathcal{O}(N)$	$\mathbf{94.5 \pm 0.3}$	59.3 ± 0.5
SignNet-8L	$\mathcal{O}(N)$	$\mathcal{O}(N)$	89.0 ± 5.2	55.2 ± 2.9
SignNet-full	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$	OOM	OOM
BasisNet	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$	OOM	OOM
SPE	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$	OOM	OOM
R-PEARL(ours)	$\mathcal{O}(MN)$	$\mathcal{O}(N)/\mathcal{O}(MN)$	$\mathbf{94.5 \pm 0.4}$	$\mathbf{60.6 \pm 0.3}$

6.1 Architectures

To generate the proposed PE, Φ is a $9-$layer message-passing GNN with batch normalization layers and skip connections. The first layer of Φ is a generalized GNN layer, as described in Eq. (3), and K can be greater than two. All the remaining layers in Φ are GIN layers (Xu et al., 2019). When $K = 2$ we omit this generalized GNN layer, and solely use GIN layers. We denote as R-PEARL the architecture with random samples, described in Eq. (8), and B-PEARL the architecture with basis vectors, described in Eq. (12).

In all experiments we evaluated our model on selected values of K ranging from 2 to 18, as well as different sample sizes ranging from 10 to 200, and selected the best model accordingly. The R-PEARL and B-PEARL encodings are fed to a GINE (Hu et al., 2020) architecture, which is a message-passing GNN that processes node and edge attributes, as well as the graph structure and PEs. More architectural and experimental details can be found in Appendices I and K.

6.2 Baselines

The baseline models for comparison are grouped into four categories: i) **GNNs without PEs**: GCN (Kipf & Welling, 2016), GIN (Xu et al., 2019); ii) **GNNs with unique identifiers**: GIN with random IDs (Xu et al., 2019; Abboud et al., 2021; Sato et al., 2019); iii) **GNNs with structural PEs**: GSN with cycles, GSN with cliques (Bouritsas et al., 2022); iv) **GNNs with eigenvector-based PEs**: SignNet, BasisNet (Lim et al.), PEG (Wang et al., 2022), SPE (Huang et al.).

In addition, we implement SignNet-8S, BasisNet-8S and SPE-8S which are variants of the full SignNet, BasisNet, and SPE models. These variants employ the eigenvectors corresponding to the 8 smallest eigenvalues of the normalized Laplacian that still need $\mathcal{O}(N^3)$ computational complexity. In SPE-8S and BasisNet-8S the memory complexity remains $\mathcal{O}(N^2)$, but in SignNet-8S it reduces from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. Furthermore, we implement SignNet-8L, utilizing only the 8 largest eigenvectors, which reduces both the memory and computational complexity to $\mathcal{O}(N)$.

6.3 Graph Classification on Social Networks

We first evaluate our architecture on graph classification tasks using the REDDIT-B (2,000 graphs, 2 classes, 429.6 average nodes) and REDDIT-M (5,000 graphs, 5 classes, 508.5 average nodes) datasets (Yanardag & Vishwanathan, 2015). Each graph represents an online discussion thread, with nodes representing different users, and edges indicating whether one user responded to another's comment. In both datasets, the task is to predict the subreddit to which a particular discussion graph belongs. To train the GNN models, we conduct 10-fold cross-validation. Table 1 summarizes the mean and standard deviation of classification accuracy over the 10 folds. We report the best performance observed during 350 epochs of training, as is the standard practice for this dataset. The results are presented in Table 1.

We observe that R-PEARL outperforms all baselines on REDDIT-M and achieves the best performance on REDDIT-B, alongside SignNet-8S, but with one to two orders of magnitude less

Table 2: logP Prediction in ZINC. B-PEARL ouperforms all the baselines both in MAE and Generalization gap. It is notable that B-PEARL achieves these results with quadratic complexity compared to the second best (SPE) that operates with cubic complexity.

PE Method	#PEs	Test MAE	Training MAE	General. Gap
No PE	N/A	0.1772 ± 0.0040	0.1509 ± 0.0086	0.0263 ± 0.0113
GCN	N/A	0.469 ± 0.002	–	–
GIN	N/A	0.209 ± 0.018	–	–
GSN with cycles	10	0.115 ± 0.012	–	–
GIN + rand id	1	0.279 ± 0.023	–	–
SignNet-8S	8	0.1034 ± 0.0056	0.0418 ± 0.0101	0.0602 ± 0.0112
SignNet	Full	0.0853 ± 0.0026	0.0349 ± 0.0078	0.0502 ± 0.0103
BasisNet-8S	8	0.1554 ± 0.0048	0.0513 ± 0.0053	0.1042 ± 0.0063
BasisNet	Full	0.1555 ± 0.0124	0.0684 ± 0.0202	0.0989 ± 0.0258
SPE-8S	8	0.0736 ± 0.0007	0.0324 ± 0.0058	0.0413 ± 0.0057
SPE	Full	0.0693 ± 0.0040	0.0334 ± 0.0054	0.0359 ± 0.0087
R-PEARL(ours)	N/A	0.0699 ± 0.002	0.0366 ± 0.006	0.0333 ± 0.007
B-PEARL(ours)	N/A	0.0644 ± 0.001	0.0290 ± 0.003	0.0353 ± 0.002

computational complexity. Notably, SignNet-full, BasisNet, and SPE are unable to handle these datasets due to their quadratic memory complexity.

6.4 Graph regression on Molecular Graphs

We also evaluate our model on the task of predicting the penalized water-octanol partition coefficient (logP) for molecules from the ZINC dataset (Irwin et al., 2012; Dwivedi et al., 2023). We use the standard split for this dataset, which entails 10,000 molecules for training, 1,000 for validation, and another 1,000 for testing. We report the mean and standard deviation of the MAE for the model achieving the highest validation accuracy, averaged over 4 different seeds. The results can be found in Table 2. We observe that B-PEARL achieves the best results, and also the best generalization gap between the competing methods. It is also notable that R-PEARL and B-PEARL also outperform all the remaining competing methods.

We conduct experiments on the DrugOOD dataset (Ji et al., 2022). The dataset evaluates models on out-of-distribution (OOD) generalization, focusing on three specific types of domain shifts: Assay, Scaffold, and Size. The Assay, Scaffold, and Size splits test the ability to generalize to different bioassays, molecules with different structures, and molecules of different sizes respectively. The results are presented in Table 3. We observe that stable methods work generally better than SignNet and BasisNet, which have reduced stability. B-PEARL improves the AUC by 9.5% compared to BasisNet, and 3.8% compared to SignNet in Size OOD generalization. B-PEARL achieves a 4.6% improvement in AUC over both SignNet and BasisNet in Scaffold OOD generalization. Furthermore, B-PEARL demonstrates an advantage over SPE in size OOD generalization. This improvement is likely attributed to the linear scaling of SPE's stability bound with graph size, compared to the square-root scaling of PEARL's stability bound, which enhances its size generalization capabilities.

6.5 Large-scale Link Prediction on Relational Databases (RelBench)

We also test the performance of the proposed PEARL on large-scale link prediction for Stack Exchange Q&A Website Database. To that end we utilize the rel-stack dataset for the relational deep learning benchmak (RelBench) Fey et al.; Robinson et al. (2024). Rel-stack is a temporal and heterogeneous graph with approximately 38 million nodes. We consider two different tasks; i) `user-post-comment`, where we predict a list of existing posts that a user will comment in the next two years, and ii) `post-post-related`, where we predict a list of existing posts that users will link a given post to in the next two years. The results for the two tasks can be found in Table 4.

The backbone model for this RelBench task a heterogeneous identity-aware GNN You et al. (2021) and all methods are trained with batch size 20. From Table 4 we observe that PEARL has an 11% benefit over the identity aware backbone model with no PE on the `user-post-comment` task and a 2% benefit on the `post-post-related` task. PEARL works similarly to SignNet-8S but with lower complexity and, and similarly to SignNet-8L on the `user-post-comment` task, but 5% better than SignNet-8L on the `post-post-related` task.

Table 3: Binding Affinity AUROC results (5 random seeds) on DrugOOD: The performance of PEARL outperforms SignNet and BasisNet and works comparably to SPE, while maintaining lower computational complexity.

Domain	PE Method	ID-Val (AUC)	ID-Test (AUC)	OOD-Val (AUC)	OOD-Test (AUC)
Assay	No PE	92.92±0.14	92.89±0.14	71.02±0.79	71.68±1.10
	PEG	92.51±0.17	92.57±0.22	70.86±0.44	71.98±0.65
	SignNet	92.26±0.21	92.43±0.27	70.16±0.56	**72.27±0.97**
	BasisNet	88.96±1.35	89.42±1.18	71.19±0.72	71.66±0.05
	SPE	92.84±0.20	92.94±0.15	71.26±0.62	**72.53±0.66**
	SPE-8S	92.36±0.18	92.62±0.10	70.71±0.47	71.72 ± 0.71
	R-PEARL(ours)	92.71±0.10	92.92±0.12	70.57±0.72	**72.24±0.30**
	B-PEARL(ours)	90.54±0.89	90.81 ±0.79	70.53±0.67	71.22±0.42
Scaffold	No PE	96.56±0.10	87.95±0.20	79.07±0.97	68.00±0.60
	PEG	95.65±0.29	86.20±0.14	79.17±0.97	69.15±0.75
	SignNet	95.48±0.34	86.73±0.56	77.81±0.70	66.43±1.06
	BasisNet	85.80±3.75	78.44±2.45	73.36±1.44	66.32±5.68
	SPE	96.32±0.28	88.12±0.41	80.03±0.58	**69.64±0.49**
	SPE-8S	96.44±0.079	87.88±0.45	79.34±0.50	68.72±0.63
	R-PEARL(ours)	96.09±0.32	88.01±0.43	78.72±0.02	69.20±1.00
	B-PEARL(ours)	96.06±0.29	87.56±0.81	79.86±0.58	**69.51±0.62**
Size	No PE	93.78±0.12	93.60±0.27	82.76±0.04	**66.04±0.70**
	PEG	92.46±0.35	92.67±0.23	82.12±0.49	66.01±0.10
	SignNet	93.30±0.43	93.20±0.39	80.67±0.50	64.03±0.70
	BasisNet	86.04±4.01	85.51±4.04	75.97±1.71	60.79±3.19
	SPE	92.46±0.35	92.67±0.23	82.12±0.49	66.02 ± 0.49
	SPE-8S	93.68±0.20	93.86±0.12	83.04±0.63	65.74 ± 2.2
	R-PEARL(ours)	93.32±0.34	93.92±0.20	82.09±0.44	65.89 ± 1.30
	B-PEARL(ours)	93.18 ± 0.45	93.29 ± 0.46	83.14 ± 0.37	**66.58 ± 0.67**

Table 4: Validation and test mean average precision (MAP) on large-scale RelBench recommendation tasks. PEARL has an 11% benefit over the backbone model with no PE on the `user-post-comment` task and a 2% benefit on the `post-post-related` task.

Task	Evaluation	No PE	SignNet-8L	SignNet-8S	B-PEARL(ours)	R-PEARL(ours)
`User-post-comment`	Val. MAP	15.20	15.33	**15.47**	15.13	15.24
	Test MAP	12.47	13.76	13.77	13.80	**13.87**
`Post-post-related`	Val. MAP	8.10	7.90	7.70	8.00	**8.40**
	Test MAP	10.73	10.39	10.86	**10.94**	10.86

7 Related Work

The works that are mostly relevant to our work can be grouped in 4 categories: i) Eigenvector-based Positional Encodings, e.g., (Dwivedi & Bresson, 2021; Rampášek et al., 2022; Kreuzer et al., 2021; Mialon et al., 2021; Feldman et al., 2022; Huang et al.; Zhang et al.); ii) Graph Neural Networks with unique node identifiers, e.g., (Loukas, 2019; Abboud et al., 2021; Sato et al., 2021; Abboud et al., 2021; Sato et al., 2021; Eliasof et al., 2023); iii) Graph Representation Learning with Structural Encodings, e.g., (Li et al., 2020; Ying et al., 2021; You et al., 2019; 2021; Dwivedi et al.; Ma et al., 2023; Kanatsoulis & Ribeiro); iv) (Wang et al., 2022; Srinivasan & Ribeiro; Murphy et al., 2018). A detailed discussion can be found in Appendix A.

8 Conclusion

In this paper, we proposed a novel framework for learnable positional encodings (PEs) that addresses key limitations in existing eigenvector-based methods, particularly in terms of stability, expressive power, scalability, and genericness. By leveraging message-passing GNNs as nonlinear mappings of eigenvectors, we designed efficient PEs that maintain permutation equivariance through the use of statistical pooling functions. Our approach not only ensures high expressiveness and stability but also significantly reduces computational complexity. Experimental results demonstrate that our method consistently outperforms lightweight eigenvector-based PEs and matches the performance of full eigenvector-based methods, all while offering substantial improvements in computational efficiency. These findings open new avenues for developing scalable, expressive, and robust graph representation techniques, paving the way for advancements in graph-based learning tasks.

REFERENCES

Ralph Abboud, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. In *IJCAI*, 2021.

Mitchell Black, Zhengchao Wan, Gal Mishne, Amir Nayyeri, and Yusu Wang. Comparing graph transformers via positional encodings. *arXiv preprint arXiv:2402.14202*, 2024.

Stéphane Boucheron, Gábor Lugosi, and Olivier Bousquet. Concentration inequalities. In *Summer school on machine learning*, pp. 208–240. Springer, 2003.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668, 2022.

Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identifications. *Combinatorica*, 12(4):389–410, 1992.

Chandler Davis and William Morton Kahan. The rotation of eigenvectors by a perturbation. iii. *SIAM Journal on Numerical Analysis*, 7(1):1–46, 1970.

Mohammed Haroon Dupty, Yanfei Dong, and Wee Sun Lee. Pf-gnn: Differentiable particle filtering based approximation of universal graph representations. In *International Conference on Learning Representations*.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. 2021.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations*.

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

Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24 (43):1–48, 2023.

Moshe Eliasof, Fabrizio Frasca, Beatrice Bevilacqua, Eran Treister, Gal Chechik, and Haggai Maron. Graph positional encoding via random feature propagation. In *International Conference on Machine Learning*, pp. 9202–9223. PMLR, 2023.

Or Feldman, Amit Boyarski, Shai Feldman, Dani Kogan, Avi Mendelson, and Chaim Baskin. Weisfeiler and leman go infinite: Spectral and combinatorial pre-colorings. *Transactions on Machine Learning Research*, 2022.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric, 2019. URL `https://arxiv.org/abs/1903.02428`.

Matthias Fey, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec. Position: Relational deep learning-graph representation learning on relational databases. In *Forty-first International Conference on Machine Learning*.

Fernando Gama, Joan Bruna, and Alejandro Ribeiro. Stability properties of graph neural networks. *IEEE Transactions on Signal Processing*, 68:5680–5695, 2020.

Simon Geisler, Yujia Li, Daniel J Mankowitz, Ali Taylan Cemgil, Stephan Günnemann, and Cosmin Paduraru. Transformers meet directed graphs. In *International Conference on Machine Learning*, pp. 11144–11172. PMLR, 2023.

Simon Geisler, Arthur Kosmala, Daniel Herbst, and Stephan Günnemann. Spatio-spectral graph neural networks. *arXiv preprint arXiv:2405.19121*, 2024.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, 2017.

Xiaoxin He, Bryan Hooi, Thomas Laurent, Adam Perold, Yann LeCun, and Xavier Bresson. A generalization of vit/mlp-mixer to graphs. In *International conference on machine learning*, pp. 12724–12745. PMLR, 2023.

W Hu, B Liu, J Gomes, M Zitnik, P Liang, V Pande, and J Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

Ningyuan Teresa Huang and Soledad Villar. A short tutorial on the weisfeiler-lehman test and its variants. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8533–8537. IEEE, 2021.

Yinan Huang, William Lu, Joshua Robinson, Yu Yang, Muhan Zhang, Stefanie Jegelka, and Pan Li. On the stability of expressive positional encodings for graphs. In *The Twelfth International Conference on Learning Representations*.

John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7): 1757–1768, 2012.

Yuanfeng Ji, Lu Zhang, Jiaxiang Wu, Bingzhe Wu, Long-Kai Huang, Tingyang Xu, Yu Rong, Lanqing Li, Jie Ren, Ding Xue, Houtim Lai, Shaoyong Xu, Jing Feng, Wei Liu, Ping Luo, Shuigeng Zhou, Junzhou Huang, Peilin Zhao, and Yatao Bian. Drugood: Out-of-distribution (ood) dataset curator and benchmark for ai-aided drug discovery – a focus on affinity prediction problems with noise annotations, 2022. URL https://arxiv.org/abs/2201.09637.

Charilaos Kanatsoulis and Alejandro Ribeiro. Counting graph substructures with graph neural networks. In *The Twelfth International Conference on Learning Representations*.

Charilaos I Kanatsoulis and Alejandro Ribeiro. Graph neural networks are more powerful than we think. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7550–7554. IEEE, 2024.

Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *Advances in Neural Information Processing Systems*, 35:14582–14595, 2022.

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

Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33:4465–4478, 2020.

Derek Lim, Joshua David Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. In *The Eleventh International Conference on Learning Representations*.

Andreas Loukas. What graph neural networks cannot learn: depth vs width. In *International Conference on Learning Representations*, 2019.

Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing. In *International Conference on Machine Learning*, pp. 23321–23337. PMLR, 2023.

Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. Graphit: Encoding graph structure in transformers, 2021. URL https://arxiv.org/abs/2106.05667.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: higher-order graph neural networks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, pp. 4602–4609, 2019.

Ryan Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational pooling for graph representations. In *International Conference on Machine Learning*, pp. 4663–4673. PMLR, 2019.

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, volume 2 of *KDD '14*, pp. 701–710. ACM, August 2014. doi: 10.1145/2623330.2623732. URL http://dx.doi.org/10.1145/2623330.2623732.

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan E Lenssen, Yiwen Yuan, Zecheng Zhang, et al. Relbench: A benchmark for deep learning on relational databases. *arXiv preprint arXiv:2407.20060*, 2024.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Approximation ratios of graph neural networks for combinatorial problems. *Advances in Neural Information Processing Systems*, 32, 2019.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pp. 333–341. SIAM, 2021.

Balasubramaniam Srinivasan and Bruno Ribeiro. On the equivalence between positional node embeddings and structural graph representations. In *International Conference on Learning Representations*.

Behrooz Tahmasebi, Derek Lim, and Stefanie Jegelka. Counting substructures with higher-order graph neural networks: Possibility and impossibility results. *arXiv preprint arXiv:2012.03174*, 2020.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. In *International Conference on Learning Representations*, 2022.

Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16, 1968.

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

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1365–1374, 2015.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In *International conference on machine learning*, pp. 7134–7143. PMLR, 2019.

Jiaxuan You, Jonathan M Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10737–10745, 2021.

Bohang Zhang, Lingxiao Zhao, and Haggai Maron. On the expressive power of spectral invariant graph neural networks. In *Forty-first International Conference on Machine Learning*.

Bohang Zhang, Shengjie Luo, Liwei Wang, and Di He. Rethinking the expressive power of gnns via graph biconnectivity. In *The Eleventh International Conference on Learning Representations*, 2023.

A Related Work

Eigenvector-based Positional Encodings: Positional encodings are a crucial component in applying transformers to graph data and further integrating structural information in graph neural networks (GNNs). A notable approach for such positional encodings (PEs) is the use of Laplacian eigenvectors. These eigenvector-based PEs have been shown to enhance performance in transformers on graph-related tasks, as demonstrated in (Dwivedi & Bresson, 2021) and (Rampášek et al., 2022). Additionally, they can be incorporated in attention mechanisms as seen in (Kreuzer et al., 2021), (Mialon et al., 2021) and (He et al., 2023). Laplacian eigenvectors can also be used to improve performance in the context of GNNs (Kim et al., 2022).

However, eigenvector-based positional encodings face challenges with stability and sign ambiguity. Small structural changes in graphs can cause significant change in eigenvectors and their corresponding positional encodings. In addition, the sign ambiguity of eigenvectors can introduce unwanted inconsistencies in these positional encodings. Works such as Lim et al. and Huang et al. address these issues by designing sign-invariant or basis-invariant models to produce these PEs, or by making the PEs more robust and stable. (Zhang et al.) introduced expressive power of spectral invariant GNNs, which are GNN architectures augmented with spectral projection matrices and provided a unified theoretical framework to analyze the previous and their proposed approach. Feldman et al. (2022) used eigenvector-based heat kernels to generate node embeddings the overcome the limitations of the WL test. Geisler et al. (2024) combine spatial and spectral graph filters in a unified GNN architecture.

Randomized Graph Neural Networks Initializing GNNs with unique node identifiers to enhance the expressive power has been first proposed by (Loukas, 2019; Abboud et al., 2021; Sato et al., 2021). In particular, (Abboud et al., 2021) and (Sato et al., 2021) used random node features as inputs to GNNs, leading to enhanced function approximation, though at the expense of permutation equivariance, a key property in graph learning. Eliasof et al. (2023) proposed a method for generating PEs in graph neural networks by leveraging random feature propagation, inspired by the power iteration method and its generalizations. The core of their approach involves concatenating several intermediate steps to compute the dominant eigenvectors of a propagation matrix. Dupty et al. proposed a randomization method that approximates the individualization-refinement technique through particle filtering. The particle filtering GNN (PF-GNN) employs a 1-WL-based initialization method, which is subsequently refined using with particle filtering sampling to overcome the 1-WL limitations.

Graph Representation Learning with Structural Encodings: Structural encodings are also important in capturing aspects of a graph's structure, such as connectivity and neighborhood information. (Li et al., 2020) uses distance PEs for GNNs (distance from an anchor node) using shortest paths and random walks. One approach is using distance-based information between nodes through methods like shortest paths or random walks, to captural structural information for transformers (Ying et al., 2021) (You et al., 2019) (You et al., 2021). Other methods learn structural PEs directly. For instance, (Dwivedi et al.) learn embeddings that are initialized with Laplacian eigenvectors or random walks. Similarly, (Ma et al., 2023) learn a linear combination of the Laplacian for creating relative PEs.

Node Embedding Methods: One foundational approach to capturing meaningful graph representations is through node embeddings. DeepWalk (Perozzi et al., 2014) and node2vec (Grover & Leskovec, 2016) are early instances of these approaches and leverage random walk strategies to learn node embeddings on graphs. Although these methods show the significance of capturing structural information, they lack expressivity and do not incorporate many learnable components.

Equivariant pooling: Similar techniques to ours have been introduced in (Wang et al., 2022), which generate PEs by applying transformations on the Laplacian. On the other hand, Kanatsoulis & Ribeiro recently analyzed the capability of GNNs to count substructures using expectation pooling functions. Srinivasan & Ribeiro explored the equivalence between node embeddings and structural representations, showing that the expectation of node embeddings can serve as structural representations of the graph, and proposed methods to sample informative node embeddings for enhanced graph representation learning. Finally, Murphy et al. (2018) investigated models of permutation-invariant functions as averages of permutation-sensitive functions applied to all reorderings of a group.

A.1 Comparing PEARL to Structural PEs

The proposed PEARL framework can provably count important substructures in any graph, such as cycles, cliques, and quasi-cliques. More importantly, it can generalize the counting function to graphs not seen during training, demonstrating the robust generalization ability of PEARL. This naturally invites comparison with methods that explicitly compute these substructures independently. Below, we summarize the key comparison points with such methods:

Expressivity: PEARL is not limited to pre-defined motifs, such as cycles or cliques. It can compute other potentially important substructures, such as dense subgraphs, chordal cycles, or combinations of motifs, that explicit counting methods might omit simply because they are not pre-specified. Notably, the number of possible motifs in a graph grows combinatorially, highlighting the flexibility and breadth of PEARL.

Complexity: Explicitly counting high-order motifs, especially at the node level, can be computationally expensive. PEARL bypasses this challenge by learning to capture these structures implicitly, making it more scalable to large and complex graphs.

Bias: Predetermining which motifs to count introduces bias into the model. For example, molecular graphs often benefit from detecting cycles, while social networks emphasize cliques or dense subgraphs. In contrast, PEARL is task-agnostic and allows the data to guide which motifs are most relevant, adapting to the specific requirements of the application. On the flip side, when the training data have small sizes, learning can benefit by specific biases that structural PEs admit.

B Proof of Proposition 3.1

Under the assumptions of Proposition 3.1 the GNN has the following recursive formula:

$$\boldsymbol{X}^{(l)} = \texttt{MLP}\left(\boldsymbol{X}^{(l-1)}, \boldsymbol{S}\boldsymbol{X}^{(l-1)}\right) = \texttt{MLP}^{(-1)}\left(\sigma\left(\sum_{k=0}^{K-1} \boldsymbol{S}^k \boldsymbol{X}^{(l-1)} \boldsymbol{H}_k^{(l)}\right)\right), \tag{13}$$

where $\texttt{MLP}^{(-1)}$ denotes the all the layers of $\texttt{MLP}$ except the first layer, and $K = 2$. We know compute the eigenvalue decomposition of $\boldsymbol{S}^k = \boldsymbol{V}\boldsymbol{\Lambda}^k\boldsymbol{V}^T$, and use some extra algebraic manipulations.

$$\boldsymbol{X}^{(l)} = \texttt{MLP}^{(-1)}\left(\sigma\left(\sum_{k=0}^{K-1}\boldsymbol{V}\boldsymbol{\Lambda}^k\boldsymbol{V}^T\boldsymbol{X}^{(l-1)}\boldsymbol{H}_k^{(l)}\right)\right) \tag{14}$$

$$= \texttt{MLP}^{(-1)}\left(\sigma\left(\sum_{k=0}^{K-1}\sum_{n=1}^{N}\lambda_n^k\boldsymbol{v}_n\boldsymbol{v}_n^T\boldsymbol{X}^{(l-1)}\boldsymbol{H}_k^{(l)}\right)\right) \tag{15}$$

$$= \texttt{MLP}^{(-1)}\left(\sigma\left(\sum_{k=0}^{K-1}\sum_{n=1}^{N}\lambda_n^k\boldsymbol{v}_n\left[\boldsymbol{v}_n^T\boldsymbol{X}^{(l-1)}[:,1],\ldots,\boldsymbol{v}_n^T\boldsymbol{X}^{(l-1)}[:,F_{l-1}]\right]\boldsymbol{H}_k^{(l)}\right)\right), \tag{16}$$

where $\boldsymbol{V}[:,n] = \boldsymbol{v}_n$ and $\boldsymbol{\Lambda}[n,n] = \lambda_n$. We now focus on the output of the first MLP layer $\boldsymbol{X}^{(l,1)}$, $\boldsymbol{X}^{(l)} = \texttt{MLP}^{(-1)}\left(\boldsymbol{X}^{(l,1)}\right)$ first layer only : Then each feature of $\boldsymbol{X}^{(l,1)}$ can be written as:

$$\boldsymbol{X}^{(l,1)}[:,f] = \sigma\left(\sum_{k=0}^{K-1}\sum_{n=1}^{N}\lambda_n^k\boldsymbol{v}_n\left[\boldsymbol{v}_n^T\boldsymbol{X}^{(l-1)}[:,1],\ldots,\boldsymbol{v}_n^T\boldsymbol{X}^{(l-1)}[:,F_{l-1}]\right]\boldsymbol{H}_k^{(l)}[:,f]\right) \tag{17}$$

$$= \sigma\left(\sum_{k=0}^{K-1}\sum_{n=1}^{N}\lambda_n^k\boldsymbol{v}_n\sum_{i=1}^{F_{l-1}}\boldsymbol{H}_k^{(l)}[i,f]\boldsymbol{v}_n^T\boldsymbol{X}^{(l-1)}[:,i]\right) \tag{18}$$

$$= \sigma\left(\sum_{n=1}^{N}\sum_{i=1}^{F_{l-1}}\sum_{k=0}^{K-1}\lambda_n^k\boldsymbol{H}_k^{(l)}[i,f] < \boldsymbol{v}_n, \boldsymbol{X}^{(l-1)}[:,i] > \boldsymbol{v}_n\right) \tag{19}$$

$$= \sigma\left(\sum_{n=1}^{N}\boldsymbol{W}[n,f]\,\boldsymbol{v}_n\right), \tag{20}$$

where:

$$\boldsymbol{W}[n,f] = \sum_{i=1}^{F_{l-1}}\sum_{k=0}^{K-1}\lambda_n^k\boldsymbol{H}_k^{(l)}[i,f] < \boldsymbol{v}_n, \boldsymbol{X}^{(l-1)}[:,i] > . \tag{21}$$

As a result $\boldsymbol{X}^{(l)} = \sigma(\boldsymbol{V}\boldsymbol{W})$. When $\boldsymbol{S}$ is not symmetric we can replace $\boldsymbol{v}_n$ to $\boldsymbol{V}^{-1}[n,:]$. This concludes the proof.

C Basis Universality of PEARL

We consider the general form of PEARL:

$$\boldsymbol{P} = \rho\left[\Phi\left(\mathcal{G},\boldsymbol{q}^{(1)}\right),\ldots,\Phi\left(\mathcal{G},\boldsymbol{q}^{(M)}\right)\right] \in \mathbb{R}^{N\times d_p}, \tag{22}$$

where ρ is a general pooling function and Φ is a message-passing GNN with skip connections. We let $\boldsymbol{q}^{(m)} = \boldsymbol{e}_m$ and $M = N$. From Proposition 3.1 we get that:

$$\boldsymbol{X}^{(l)} = \texttt{MLP}(\boldsymbol{V}) = \texttt{MLP}^{(-1)}(\sigma(\boldsymbol{V}\boldsymbol{W})) \tag{23}$$

$$\boldsymbol{W}[n,f] = \sum_{i=1}^{F_{l-1}}\sum_{k=0}^{K-1}\lambda_n^k\boldsymbol{H}_k^{(l)}[i,f]\langle\boldsymbol{v}_n, \boldsymbol{X}^{(l-1)}[:,i]\rangle, \tag{24}$$

We ommit the nonlinearities from the GNN and for $\boldsymbol{X}^{(0)} = \boldsymbol{e}_m$ we get:

$$\boldsymbol{X}^{(K)} = \boldsymbol{V}\boldsymbol{W}, \quad \boldsymbol{W}[n,f] = \langle\boldsymbol{v}_n,\boldsymbol{e}_m\rangle\sum_{k=0}^{K-1}\boldsymbol{h}_k[f]\lambda_n^k, \tag{25}$$

As a result $\boldsymbol{W}[n,f]$ is a polynomial on the eigenvalues $\tilde{h}_f(\lambda_n) = \sum_{k=0}^{K-1}\boldsymbol{h}_k[f]\lambda_n^k$ scaled by $\langle\boldsymbol{v}_n,\boldsymbol{e}_m\rangle$. We will then use the following lemma.

Lemma C.1 *Let $\mathcal{G}$ be a graph with N nodes and GSO $\boldsymbol{S} \in \mathbb{R}^{N \times N}$. Also let $\mathcal{S} = \{\{\lambda_1, \ldots, \lambda_N\}\}$ be the multiset of eigenvalues of $\boldsymbol{S}$; $\mathcal{S}$ can have repeated elements (eigenvalues). Also, let $\mathcal{M} = \{\mu_1, \ldots, \mu_q\}$ be the ordered set of all distinct (non-repeated) eigenvalues of $\boldsymbol{S}$. We can always design poynomial filter such that:*

$$\tilde{h}(\lambda) = \begin{cases} 1, \text{ if } \lambda = \mu_f \\ 0, \text{ if } \lambda = \mu_f \neq \mu_i \end{cases} \tag{26}$$

Proof: Let

$$\begin{bmatrix} \tilde{h}(\mu_1) \\ \tilde{h}(\mu_2) \\ \vdots \\ \tilde{h}(\mu_q) \end{bmatrix} = \begin{bmatrix} 1 \; \mu_1 \; \mu_1^2 \ldots \mu_1^{K-1} \\ 1 \; \mu_2 \; \mu_2^2 \ldots \mu_2^{K-1} \\ \vdots \\ 1 \; \mu_q \; \mu_q^2 \ldots \mu_q^{K-1} \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{K-1} \end{bmatrix} = \boldsymbol{W}\boldsymbol{h} \tag{27}$$

$\boldsymbol{W}$ is a Vandermonde matrix and when $K = q$ the determinant of $\boldsymbol{W}$ takes the form:

$$det(\boldsymbol{W}) = \prod_{1 \leq i < j \leq q} (\mu_i - \mu_j) \tag{28}$$

Since the values μ_i are distinct, $\boldsymbol{W}$ has full column rank and there exists a polynomial $\tilde{h}$ with unique parameters $\boldsymbol{h} = \boldsymbol{W}^{-1}\boldsymbol{e}_i$ such that $\tilde{h}(\lambda) = 1$ if $\lambda = \mu_f$, and $\tilde{h}(\lambda) = 0$ if $\lambda = \mu_j \neq \mu_f$.

Using Lemma C.1, we can design $\tilde{h}_f(\lambda_n) = \sum_{k=0}^{K-1} \boldsymbol{h}_k[f]\lambda_n^k$ such that:

$$\tilde{h}_f(\lambda_n) = \begin{cases} 1, \text{ if } \lambda_n = \mu_f \\ 0, \text{ if } \lambda_n = \mu_j \neq \mu_f \end{cases} \tag{29}$$

Under this parametrization, $\boldsymbol{X}^{(K)}$ takes the form:

$$\boldsymbol{X}^{(K)} = \left[\boldsymbol{V}_{\mu_1}\boldsymbol{V}_{\mu_1}^T\boldsymbol{e}_m, \ldots, \boldsymbol{V}_{\mu_q}\boldsymbol{V}_{\mu_q}^T\boldsymbol{e}_m\right] \in \mathbb{R}^{N \times q}, \tag{30}$$

where $\boldsymbol{V}_{\mu_f}$ is the eigenspace (orthogonal space of the eigenvectors) corresponding to eigenvalue μ_f. Since we independently feed $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_N$ to the PEARL architecture, we will have N output samples for each output feature, i.e., N samples for $\boldsymbol{X}^{(K)}[:, f]$. We will represent the $m-$th sample as $\boldsymbol{X}^{(K)}[:, f, m]$ and for the $f-$th output feature will have the following samples:

$$\boldsymbol{X}^{(K)}[:, f, :] = \left[\boldsymbol{V}_{\mu_f}\boldsymbol{V}_{\mu_f}^T\boldsymbol{e}_1, \ldots, \boldsymbol{V}_{\mu_f}\boldsymbol{V}_{\mu_f}^T\boldsymbol{e}_N\right] = \boldsymbol{V}_{\mu_f}\boldsymbol{V}_{\mu_f}^T, \tag{31}$$

We process the output samples of each feature via a pooling function ρ, to get the final output embedding as:

$$\boldsymbol{Y} = \rho\left(\boldsymbol{V}_{\mu_1}\boldsymbol{V}_{\mu_1}^T, \ldots, \boldsymbol{V}_{\mu_q}\boldsymbol{V}_{\mu_q}^T\right). \tag{32}$$

We can choose ρ to be a different function for each feature i.e.,

$$\boldsymbol{Y} = \rho\left(g^{(1)}\left(\boldsymbol{V}_{\mu_1}\boldsymbol{V}_{\mu_1}^T\right), \ldots, g^{(q)}\left(\boldsymbol{V}_{\mu_q}\boldsymbol{V}_{\mu_q}^T\right)\right). \tag{33}$$

Equation (33) is the definition of BasisNet (Lim et al.). BasisNet universally approximates all continuous basis invariant function, which proves that PEARL is also a universal approximator of basis invariant functions.

D VERTEX AND FREQUENCY DOMAIN ANALYSIS OF RPE

Let the input to the GNN encoder $\boldsymbol{q} = [q_1, \ldots, q_N]$ be a set of N i.i.d. random variables such that $\mathbb{E}[q_i] = 0$ and $\mathbb{E}[q_i^p] = 1$ for $p \geq 2$. As shown in Eq. (48) $\boldsymbol{q}$ is processed by a set of functions:

$$\boldsymbol{y} = \sigma(\boldsymbol{z}) = \sigma\left(\sum_{k=0}^{K-1} h_k\boldsymbol{q}\right) = \sigma\left(\boldsymbol{H}(\boldsymbol{S})\,\boldsymbol{q}\right). \tag{34}$$

Now we assume that the pointwise nonlinearity is analytic and expand it as a Taylor series:

$$\boldsymbol{y} = \sigma(\boldsymbol{z}) = \sum_{n=0}^{\infty} \beta_n \boldsymbol{z}^n = \sum_{n=0}^{\infty} \beta_n \left(\boldsymbol{H}(\boldsymbol{S})\boldsymbol{q}\right)^n, \tag{35}$$

where $\beta_n = \frac{\sigma^{(n)}(0)}{n!}$. If we only use one layer, each feature of our RPEs will be:

$$\boldsymbol{p} = \mathbb{E}[\boldsymbol{y}] = \mathbb{E}[\sigma(\boldsymbol{z})] = \sum_{n=0}^{\infty} \beta_n \mathbb{E}[\boldsymbol{z}^n] = \sum_{n=0}^{\infty} \beta_n \mathbb{E}\left[\left(\boldsymbol{H}(\boldsymbol{S})\boldsymbol{q}\right)^n\right] \tag{36}$$

$$= \sum_{n=0}^{\infty} \beta_n \underbrace{\boldsymbol{H}(\boldsymbol{S}) * \cdots * \boldsymbol{H}(\boldsymbol{S})}_{n \text{ times}} \mathbf{1} = \sum_{n=0}^{\infty} \sum_{i_1, i_2, \ldots, i_n} \beta_n h_{i_1} \ldots h_{i_n} \left(\boldsymbol{S}^{i_1} * \cdots * \boldsymbol{S}^{i_n}\right) \mathbf{1} \tag{37}$$

where $*$ represents the Hadamard product. As a result, the produced PE is a linear combination of the following features $\left(\boldsymbol{S}^{i_1} * \cdots * \boldsymbol{S}^{i_n}\right)\mathbf{1}$. Using more layers to produce the proposed PEs yields more complex functions. As we proved in Proposition 3.1, a GNN operates as a nonlinear function of eigenvectors. To exactly analyze the proposed PEs as functions of eigenvectors let $\boldsymbol{S} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T$, be the eigendecomposition of the GSO $\boldsymbol{V}$. Then we can show that:

$$\mathbb{E}[\boldsymbol{z}^n] = \sum_{i_1, \ldots, i_m = 0}^{K} h_{i_1, \ldots, i_n} \left(\boldsymbol{V}^{i_1^T} \odot \cdots \odot \boldsymbol{V}^{i_n^T}\right)^T \left(\boldsymbol{\Lambda}^{i_1^T} \otimes \cdots \otimes \boldsymbol{\Lambda}^{i_n^T}\right) \left(\boldsymbol{V}^{i_1^T} \odot \cdots \odot \boldsymbol{V}^{i_n^T}\right) \mathbf{1}, \tag{38}$$

where $\otimes$ represents the Kronecker product and $\odot$ represents the Khatri-Rao product (columnwise Kronecker). The Equation in (38) is a linear combination of eigenvector "monomials". In other words Eq. (38) instantiates Hadamard products of different eigenvector combinations and linearly combines them.

E Sample Complexity

To characterize the sample complexity of our approach we will use this version of Chebychef's inequality (Boucheron et al., 2003) as:

$$P\left(\frac{1}{M}\left|\sum_{m=1}^{M}\left(\boldsymbol{P}^{(m)} - \mathbb{E}[\Phi(\mathcal{G}, \boldsymbol{q})]\right)\right| \geq \epsilon\right) \leq \frac{\text{var}\left(\Phi(\mathcal{G}, \boldsymbol{q})\right)}{M \cdot \epsilon^2}. \tag{39}$$

To establish a bound for the variance of the output $\Phi(\mathcal{G}, \boldsymbol{q})$, we begin by analyzing how pointwise nonlinearity affects the variance of a random variable. Let X be a random variable with variance $\text{Var}(X)$, and let σ be a Lipschitz continuous function with Lipschitz constant C_σ. Our goal is to examine the impact of applying σ to X, specifically focusing on how it influences the variance of the transformed variable $\sigma(X)$.

E.1 Effect of pointwise activation to the variance of a random variable

Since σ is Lipschitz continuous with constant C_σ, for any values of X and $\mathbb{E}[X]$, we can apply the Lipschitz condition:

$$|\sigma(X) - \sigma(\mathbb{E}[X])| \leq C_\sigma |X - \mathbb{E}[X]|.$$

Taking squares on both sides:

$$\left(\sigma(X) - \sigma(\mathbb{E}[X])\right)^2 \leq C_\sigma^2 \left(X - \mathbb{E}[X]\right)^2.$$

Now, take the expectation of both sides:

$$\mathbb{E}[\left(\sigma(X) - \sigma(\mathbb{E}[X])\right)^2] \leq C_\sigma^2 \mathbb{E}[\left(X - \mathbb{E}[X]\right)^2].$$

Since $\mathbb{E}[\left(X - \mathbb{E}[X]\right)^2] = \text{Var}(X)$, this simplifies to:

$$\mathbb{E}[\left(\sigma(X) - \sigma(\mathbb{E}[X])\right)^2] \leq C_\sigma^2 \text{Var}(X).$$

Now let's work on the left-hand side of the previous equation:

$$\mathbb{E}[(\sigma(X) - \sigma(\mathbb{E}[X]))^2] = \mathbb{E}\left[(\sigma(X) - \mathbb{E}[\sigma(X)] + \mathbb{E}[\sigma(X)] - \sigma(\mathbb{E}[X]))^2\right] \tag{40}$$

$$= \mathbb{E}[(\sigma(X) - \mathbb{E}[\sigma(X)])^2] + (\mathbb{E}[\sigma(X)] - \sigma(\mathbb{E}[X]))^2 \tag{41}$$

$$+ 2\mathbb{E}[< (\sigma(X) - \mathbb{E}[\sigma(X)]), (\mathbb{E}[\sigma(X)] - \sigma(\mathbb{E}[X])) >] \tag{42}$$

$$= \mathbb{E}[(\sigma(X) - \mathbb{E}[\sigma(X)])^2] + (\mathbb{E}[\sigma(X)] - \sigma(\mathbb{E}[X]))^2 \tag{43}$$

$$= \text{Var}(\sigma(X)) + (\mathbb{E}[\sigma(X)] - \sigma(\mathbb{E}[X]))^2, \tag{44}$$

Now let $\mu = (\mathbb{E}[\sigma(X)] - \sigma(\mathbb{E}[X]))$, then the variance of $\sigma(X)$ is bounded by:

$$\text{Var}(\sigma(X)) \leq C_\sigma^2 \text{Var}(X) - \mu^2 \leq C_\sigma^2 \text{Var}(X). \tag{45}$$

This shows that the Lipschitz constant C_σ acts as a scaling factor on the variance of the random variable. If C_σ is large, the variance of $\sigma(X)$ can be significantly larger, and if C_σ is small, it can shrink the variance accordingly. For the majority of nonlinearities used in deep learning, as ReLU, sigmoid, and hyperbolic tangent, $C_\sigma = 1$, and $\text{Var}(\sigma(X)) \leq \text{Var}(X)$.

E.2 Effect of Graph Convolution to the variance of a random node signal

The next step is to study the effect of graph convolution (linear message-passing) operations to a set of node features. In particular, let $\boldsymbol{X}^{(l)} \in \mathbb{R}^{N \times F_{l-1}}$ be the node input to the $l - th$ GNN layer. Then we define $\boldsymbol{Z}^{(l)} \in \mathbb{R}^{N \times F_l}$ as:

$$\boldsymbol{X}^{(l)} = \sigma\left(\boldsymbol{Z}^{(l)}\right), \; \boldsymbol{Z}^{(l)} = \sum_{k=0}^{K-1} \boldsymbol{S}^k \boldsymbol{X}^{(l-1)} \boldsymbol{H}_k \tag{46}$$

After some algebraic manipulations, we can see that:

$$\boldsymbol{Z}^{(l)} = \sum_{k=0}^{K-1} \boldsymbol{S}^k \sum_{f=1}^{F_{l-1}} \boldsymbol{X}^{(l-1)}[:, f] \boldsymbol{H}_k[f, :]^T = \sum_{f=1}^{F_{l-1}} \sum_{k=0}^{K-1} \boldsymbol{S}^k \boldsymbol{X}^{(l-1)}[:, f] \boldsymbol{H}_k[f, :]^T, \tag{47}$$

and each feature of $\boldsymbol{Z}^{(l)}$ can be cast as:

$$\boldsymbol{Z}^{(l)}[:, d] = \sum_{f=1}^{F_{l-1}} \sum_{k=0}^{K-1} \boldsymbol{H}_k[f, d] \boldsymbol{S}^k \boldsymbol{X}^{(l-1)}[:, f], \;\; d \in \{1, \ldots, F_l\}. \tag{48}$$

The above equation implies that each feature $\boldsymbol{Z}^{(l)}[:, d]$ is generated by a summation over F_{l-1} features of type:

$$\boldsymbol{z} = \sum_{k=0}^{K-1} \boldsymbol{h}_k \boldsymbol{S}^k \boldsymbol{x} = \boldsymbol{H}(\boldsymbol{S}) \boldsymbol{x} \tag{49}$$

We assume that norm of $\boldsymbol{H}(\boldsymbol{S}) = \sum_k h_k \boldsymbol{S}^k$ is bounded, i.e., $\|\boldsymbol{H}(S)\| \leq \beta$.

As a result, we will first analyze the variance of $\boldsymbol{z}$ when the input $\boldsymbol{x}$ has covariance matrix:

$$\mathbb{E}\left[(\boldsymbol{x} - \mathbb{E}[x])(\boldsymbol{x} - \mathbb{E}[x])^T\right] = \boldsymbol{C} \tag{50}$$

The covariance of $\boldsymbol{z}$ is written as:

$$\mathbb{E}\left[(\boldsymbol{z} - \mathbb{E}[z])(\boldsymbol{z} - \mathbb{E}[z])^T\right] = \boldsymbol{H}(\boldsymbol{S}) \boldsymbol{Q} \boldsymbol{H}(\boldsymbol{S}) = \sum_{k=0}^{K-1} h_k \boldsymbol{S}^k \boldsymbol{C} \sum_{m=0}^{K-1} h_m \boldsymbol{S}^m \tag{51}$$

$$= \sum_{k=0}^{K-1} \sum_{m=0}^{K-1} h_k h_m \boldsymbol{S}^k \boldsymbol{C} \boldsymbol{S}^m \tag{52}$$

and the variance for each individual variable $\boldsymbol{z}[i]$ is:

$$\text{var}\left(\boldsymbol{z}\left[i\right]\right) = \sum_{k=0}^{K-1}\sum_{l=0}^{K-1} h_k h_l \boldsymbol{S}^k[i,:]^T \boldsymbol{Q} \boldsymbol{S}^l[:,] \tag{53}$$

$$= \sum_{k=0}^{K-1}\sum_{l=0}^{K-1} h_k h_l \sum_{m\in\mathcal{N}_i^{(k)}}\sum_{n\in\mathcal{N}_i^{(l)}} \boldsymbol{S}^k[i,m]\boldsymbol{S}^l[i,n]\text{cov}\left(\boldsymbol{x}\left[m\right],\boldsymbol{x}\left[n\right]\right) \tag{54}$$

$$\leq \sum_{k=0}^{K-1}\sum_{l=0}^{K-1} h_k h_l \sum_{m\in\mathcal{N}_i^{(k)}}\sum_{n\in\mathcal{N}_i^{(l)}} \boldsymbol{S}^k[i,m]\boldsymbol{S}^l[i,n]\left|\text{cov}\left(\boldsymbol{x}\left[m\right],\boldsymbol{x}\left[n\right]\right)\right| \tag{55}$$

$$\leq \sum_{k=0}^{K-1}\sum_{l=0}^{K-1} h_k h_l \sum_{m\in\mathcal{N}_i^{(k)}}\sum_{n\in\mathcal{N}_i^{(l)}} \boldsymbol{S}^k[i,m]\boldsymbol{S}^l[i,n]\max_i\left(\text{var}\left(\boldsymbol{x}\left[i\right]\right)\right) \tag{56}$$

$$\leq \sum_{k=0}^{K-1}\sum_{l=0}^{K-1} h_k h_l \text{deg}_{\max}^k \text{deg}_{\max}^l \max_i\left(\text{var}\left(\boldsymbol{x}\left[i\right]\right)\right) \tag{57}$$

$$\leq \sum_{k=0}^{K-1} h_k \text{deg}_{\max}^k \sum_{l=0}^{K-1} h_l \text{deg}_{\max}^l \max_i\left(\text{var}\left(\boldsymbol{x}\left[i\right]\right)\right) \leq \beta^2 \max_i\left(\text{var}\left(\boldsymbol{x}\left[i\right]\right)\right), \tag{58}$$

where $\text{deg}_{\max}^k$ is the maximum degree of $\boldsymbol{S}^k$, and is equal to $\text{deg}_{\max}^k = 1$, when $\boldsymbol{S}$ is the normalized adjacency matrix or the random walk matrix. The inequality in (56) comes from the Cauchy-Schwartz inequality, the inequality in (57) comes from the definition of $\boldsymbol{S}^k$, and the last inquality in (58) is due the boundedness of the operator $\boldsymbol{H}\left(S\right)$.

Overall,

$$\text{var}\left(\boldsymbol{z}\left[i\right]\right) \leq \beta^2 \max_i\left(\text{var}\left(\boldsymbol{x}\left[i\right]\right)\right). \tag{59}$$

The final step is to analyze the the variance of a random variable that is a sum of dependent random variables, $\boldsymbol{z}[i] = \sum_{f=1}^{F_{l-1}} \boldsymbol{z}_f[i]$. Then:

$$\text{var}\left(\boldsymbol{z}\left[i\right]\right) = \mathbb{E}\left[\left(\sum_{f=1}^{F_{l-1}} \boldsymbol{z}_f[i]\right)^2\right] = \sum_{f=1}^{F_{l-1}}\sum_{g=1}^{F_{l-1}} \mathbb{E}\left[\boldsymbol{z}_f[i],\boldsymbol{z}_g[i]\right] \leq \sum_{f=1}^{F_{l-1}}\sum_{g=1}^{F_{l-1}} \left|\mathbb{E}\left[\boldsymbol{z}_f[i],\boldsymbol{z}_g[i]\right]\right| \tag{60}$$

$$\leq F_{l-1}^2 \max_f\left(\text{var}\left(\boldsymbol{z}_f\left[i\right]\right)\right), \tag{61}$$

where the last inequality in (56) comes from the Cauchy-Schwartz inequality. which is quadratic with respect to the length of the GNN layer. Combining Eq. (45), (59), and (60 we conclude that:

$$\text{var}\left[\boldsymbol{X}^{(l)}\right] \leq C_\sigma^2 \beta^2 F_{l-1}^2 \max\left(\text{var}\left[\boldsymbol{X}^{(l-1)}\right]\right), \tag{62}$$

If we assume the $F_l = F$ for all hidden layers, we get that:

$$\text{var}\left[\boldsymbol{X}^{(L)}\right] \leq \left(C_\sigma \beta F\right)^{2L} \max\left(\text{var}\left[\boldsymbol{X}\right]\right), \tag{63}$$

In our proposed approach $\max\left(\text{var}\left[\boldsymbol{X}\right]\right) = 1$. If we further assume that $C_\sigma = 1$, which is usually the case in practice, and that $\beta = 1/F$, which means that the magnitude of trainable parameters is inversely proportional to the number of hidden dimensions in each layer, we get that:

$$\text{var}\left[\boldsymbol{X}^{(L)}\right] \leq 1. \tag{64}$$

We can now derive the following Theorem

Proposition E.1 (Sample Complexity) *Let $\boldsymbol{P}$ denote the output of the architecture described in Eq. (8), for a graph $\mathcal{G}$ with i.i.d. initial node attributes with unit variance. Also let Φ be an $L-$layer GNN*

described by Eq. (3), with F hidden dimensions at each layer. If $C_\sigma = 1$ and $\beta = 1/F$, the number of samples required such that:

$$\left| \frac{1}{M} \sum_{m=1}^{M} \Phi\left(\mathcal{G}, \boldsymbol{q}^{(m)}\right) - \mathbb{E}\left[\Phi\left(\mathcal{G}, \boldsymbol{q}\right)\right] \right| < \epsilon, \text{ with probability at least } 1-\delta, \tag{65}$$

satisfies:

$$M \leq \frac{1}{\delta \cdot \epsilon^2}. \tag{66}$$

F Proof of Corollary 4.4

To prove Corollary 4.4, we assume that the pointwise nonlinearities σ are elementwise power functions, i.e., $\sigma(\cdot) = (\cdot)^p$ for integer values of $p \geq 2$. Using this assumption, we can apply Theorem K.1 from (Kanatsoulis & Ribeiro) to establish the result. For approximate results, classical smooth nonlinearities such as the hyperbolic tangent, the sigmoid, or the Swish function can be employed and analyzed via their Taylor series expansion around zero:

$$\sigma(x) = \sum_{p=0}^{K-1} \frac{\sigma^{(p)}(0)}{p!} x^p, \tag{67}$$

where $\sigma^{(p)}$ represents the p-th derivative of $\sigma(x)$ evaluated at 0. It is straightforward to observe that elementwise power functions appear in this expansion, enabling approximate cycle counting. In any case, the proposed PEs will contain rich information relevant to cycle counts.

G Stability Analysis

Let $\tilde{\mathcal{G}}$ be a perturbed version of graph $\mathcal{G}$, with GSOs $\tilde{\boldsymbol{S}}$ and $\boldsymbol{S}$ respectively. We consider two perturbation models, i.e., additive and relative perturbation:

$$\text{Additive perturbation model:} \quad \tilde{\boldsymbol{S}} = \boldsymbol{S} + \boldsymbol{E} \tag{68}$$

$$\text{Relative perturbation model:} \quad \tilde{\boldsymbol{S}} = \boldsymbol{S}\boldsymbol{E} + \boldsymbol{S}\boldsymbol{E} \tag{69}$$

To measure the distance between $\tilde{\boldsymbol{S}}$ and $\boldsymbol{S}$, as well as the GNN outputs when the input graphs are perturbed versions of each other we define the distance modulo permutation:

Definition G.1 (Linear operator distance modulo permutation) *(Gama et al., 2020) Given linear operators $\boldsymbol{A}$ and $\tilde{\boldsymbol{A}}$ we define the operator distance modulo permutation as*

$$\left\|\boldsymbol{A} - \tilde{\boldsymbol{A}}\right\|_{\mathcal{P}} = \min_{\boldsymbol{\Pi}} \max_{\boldsymbol{x}:\|\boldsymbol{x}\|=1} \left\|\boldsymbol{\Pi}^T(\boldsymbol{A}\boldsymbol{x}) - \tilde{\boldsymbol{A}}(\boldsymbol{\Pi}^T\boldsymbol{x})\right\|, \tag{70}$$

where $\boldsymbol{\Pi}$ is a permutation matrix.

G.1 Lipschitz and Integral Lipschitz Filters

Next, we need to define the notion of Lipschitz and Integral Lipschitz filters. First, we note that graph filters are pointwise operators in the graph frequency domain, i.e.,

$$\boldsymbol{H}(\boldsymbol{S}) = \sum_{k=0}^{K-1} h_k \boldsymbol{S}^k = \sum_{k=0}^{K-1} h_k \boldsymbol{V}\Lambda^k\boldsymbol{V}^T = \boldsymbol{V}\left(\sum_{k=0}^{K-1} h_k \boldsymbol{\Lambda}^k\right)\boldsymbol{V}^T. \tag{71}$$

We can therefore define the graph frequency response of the filter as:

$$h(\lambda) = \sum_{k=0}^{K-1} h_k \lambda^k. \tag{72}$$

To continue our analysis we define the following filter types.

Definition G.2 (Lipschitz Filter) *(Gama et al., 2020) Given a filter $\boldsymbol{h} = \{h_k\}_{k=0}^{K-1}$ its frequency response $h(\lambda)$ is given by equation 72. We say the filter is Lipschitz if there exists a constant $C > 0$ such that for all λ_1 and λ_2,*

$$\big|h(\lambda_2) - h(\lambda_1)\big| \leq C\big|\lambda_2 - \lambda_1\big|. \tag{73}$$

Definition G.3 (Integral Lipschitz Filter) *(Gama et al., 2020) Given a filter $\boldsymbol{h} = \{h_k\}_{k=0}^{K-1}$ its frequency response $h(\lambda)$ is given by equation 72. We say the filter is integral Lipschitz if there exists a constant $C > 0$ such that for all λ_1 and λ_2,*

$$|h(\lambda_2) - h(\lambda_1)| \;\leq\; C\,\frac{|\lambda_2 - \lambda_1|}{|\lambda_1 + \lambda_2|\,/\,2}\,. \tag{74}$$

G.2 Stability bounds for random and basis PEs

We can use any stability bounds for GNNs. To see that let:

$$\left\|\Phi\left(\mathcal{G},\cdot\right)[:,f] - \Phi\left(\tilde{\mathcal{G}},\cdot\right)[:,f]\right\|_{\mathcal{P}} \leq \Gamma \tag{75}$$

Then

$$\begin{aligned}\left\|\frac{1}{M}\sum_{m=1}^{M}\Phi\left(\mathcal{G},\cdot\right)[:,f] - \frac{1}{M}\sum_{m=1}^{M}\Phi\left(\tilde{\mathcal{G}},\cdot\right)[:,f]\right\|_{\mathcal{P}} &\leq \frac{1}{M}\left\|\sum_{m=1}^{M}\Phi\left(\mathcal{G},\cdot\right)[:,f] - \Phi\left(\tilde{\mathcal{G}},\cdot\right)[:,f]\right\|_{\mathcal{P}} \\ &\leq \frac{1}{M}\sum_{m=1}^{M}\left\|\Phi\left(\mathcal{G},\cdot\right)[:,f] - \Phi\left(\tilde{\mathcal{G}},\cdot\right)[:,f]\right\|_{\mathcal{P}} \\ &\leq \Gamma \end{aligned} \tag{76}$$

Using the previous definitions and Eq. (76) we can now use the analysis in (Gama et al., 2020) to establish the stability of the proposed PEs.

Proposition G.1 (Stability to additive perturbations) *Let $\tilde{\mathcal{G}}$ be a perturbed version of $\mathcal{G}$ such that $\tilde{\boldsymbol{S}} = \boldsymbol{S} + \boldsymbol{E}$ with $\|\boldsymbol{E}\| \leq \varepsilon$. Let Φ be an $L-$layer GNN described by Eq. (3), where each layer consists of Lipschitz filters with constant C. Under assumptions 4.1 and 4.2 with $C_\sigma = 1$ and $\beta = 1/F$, it holds that:*

$$\left\|\frac{1}{M}\sum_{m=1}^{M}\Phi\left(\mathcal{G},\cdot\right)[:,f] - \frac{1}{M}\sum_{m=1}^{M}\Phi\left(\tilde{\mathcal{G}},\cdot\right)[:,f]\right\|_{\mathcal{P}} \leq \left(1 + 8\sqrt{N}\right)L\varepsilon + \mathcal{O}(\varepsilon^2) \tag{77}$$

Proposition G.2 (Stability to relative perturbations) *Let $\tilde{\mathcal{G}}$ be a perturbed version of $\mathcal{G}$ such that $\tilde{\boldsymbol{S}} = \boldsymbol{S} + \boldsymbol{S}\boldsymbol{E} + \boldsymbol{E}\boldsymbol{S}$ with $\|\boldsymbol{E}\|_{\mathcal{P}} \leq \varepsilon$. Let Φ be an $L-$layer GNN described by Eq. (3), where each layer consists of Integral Lipschitz filters with constant C. Under assumptions 4.1 and 4.2 with $C_\sigma = 1$ and $\beta = 1/F$, it holds that:*

$$\left\|\frac{1}{M}\sum_{m=1}^{M}\Phi\left(\mathcal{G},\cdot\right)[:,f] - \frac{1}{M}\sum_{m=1}^{M}\Phi\left(\tilde{\mathcal{G}},\cdot\right)[:,f]\right\|_{\mathcal{P}} \leq 2\left(1 + 8\sqrt{N}\right)L\varepsilon + \mathcal{O}(\varepsilon^2) \tag{78}$$

H Spectral filters with Graph Filters

The suggested implementation in (Huang et al.) is:

$$\begin{aligned}\text{SPE}\left(\boldsymbol{V},\boldsymbol{\Lambda}\right) &= \sum_{n=0}^{N-1}\rho\left(\left[\boldsymbol{V}\text{diag}\left(\alpha_1\left(\boldsymbol{\Lambda}\right)\right)\boldsymbol{V}[n]^T,\ldots,\boldsymbol{V}\text{diag}\left(\alpha_M\left(\boldsymbol{\Lambda}\right)\right)\boldsymbol{V}[n]^T\right]\right) \qquad (79)\\ &= \sum_{n=0}^{N-1}\rho\left(\left[\boldsymbol{V}\text{diag}\left(\alpha_1\left(\boldsymbol{\Lambda}\right)\right)\boldsymbol{V}^T\boldsymbol{e}_n,\ldots,\boldsymbol{V}\text{diag}\left(\alpha_M\left(\boldsymbol{\Lambda}\right)\right)\boldsymbol{V}^T\boldsymbol{e}_n\right]\right), \qquad (80)\end{aligned}$$

where ρ represents multiple GIN layers. If we assume that α_m are analytic element wise functions, then we can take the taylor series expansion and represent α_m as a polynomial. Then SPE can be cast as:

$$\text{SPE}\left(\boldsymbol{V}, \boldsymbol{\Lambda}\right) = \sum_{n=0}^{N-1} \rho\left(\left[\boldsymbol{V}\text{diag}\left(\sum_{k=0}^{K-1} h_k^1 \boldsymbol{\Lambda}^k\right)\boldsymbol{V}^T\boldsymbol{e}_n, \ldots, \boldsymbol{V}\text{diag}\left(\sum_{k=0}^{K-1} h_k^M \boldsymbol{\Lambda}^k\right)\boldsymbol{V}^T\boldsymbol{e}_n\right]\right) \tag{81}$$

$$= \sum_{n=0}^{N-1} \rho\left(\left[\boldsymbol{V}\sum_{k=0}^{K-1} h_k^1 \boldsymbol{\Lambda}^k\boldsymbol{V}^T\boldsymbol{e}_n, \ldots, \boldsymbol{V}\sum_{k=0}^{K-1} h_k^M \boldsymbol{\Lambda}^k\boldsymbol{V}^T\boldsymbol{e}_n\right]\right) \tag{82}$$

$$= \sum_{n=0}^{N-1} \rho\left(\left[\sum_{k=0}^{K-1} h_k^1 \boldsymbol{V}\boldsymbol{\Lambda}^k\boldsymbol{V}^T\boldsymbol{e}_n, \ldots, \sum_{k=0}^{K-1} h_k^M \boldsymbol{V}\boldsymbol{\Lambda}^k\boldsymbol{V}^T\boldsymbol{e}_n\right]\right) \tag{83}$$

$$= \sum_{n=0}^{N-1} \rho\left(\left[\sum_{k=0}^{K-1} h_k^1 \boldsymbol{A}^k\boldsymbol{e}_n, \ldots, \sum_{k=0}^{K-1} h_k^M \boldsymbol{A}^k\boldsymbol{e}_n\right]\right) = \sum_{n=0}^{N-1} \rho\left(\sum_{k=0}^{K-1} \boldsymbol{A}^k\boldsymbol{e}_n\boldsymbol{h}_k^T\right) \tag{84}$$

The expression in Eq. (84) coincides with the B-PEARL architecture, which concludes our proof.

I Implementation Details

All results for the SPE, SignNet, and BasisNet models using only 8 eigenvectors were either sourced from their original papers, when available, or obtained by retraining the original models with 8 eigenvectors corresponding to the 8 largest or smallest eigenvalues. All other baseline results were sourced from their original papers. For both the R-PEARL and B-PEARL models, batch normalization is applied within the Φ layers. Additionally, when $K > 2$, the output of the first Φ layer is passed through a shallow MLP consisting of 1 or 2 layers before continuing through the remaining layers.

For the REDDIT datasets, we use R-PEARL with 30 samples and $K = 2$, omitting the the first layer described in Eq. (3). Both SignNet and R-PEARL use 4 GIN layers with batch normalization to generate the positional encodings, followed by a base model consisting of 6 additional GIN layers. In R-PEARL, skip connections are applied across those 4 GIN layers, followed by a linear layer at the end. SignNet uses residual connections instead, and also uses MLP encoders for the eigenvectors, as well as a Set Transformer Lee et al. (2019). For both models, we use a batch size of 70 and 100 on REDDIT-BINARY and REDDIT-MULTI respectively.

On the ZINC datasets, R-PEARL, B-PEARL, and SPE use a batch size of 128. The base model for each is a 4-layer GINE. Similar to the SPE model, we inject the original positional encoding into every layer by passing it through an MLP and adding it to the layer's input. Notably, our model employs 8 GIN layers with 40 hidden units for Φ, whereas SPE uses an 8-layer GIN with 128 hidden units, in addition to 3 MLPs. For R-PEARL we use 50-120 samples and $K = 12$, while for B-PEARL we use $K = 4$.

For the DrugOOD datasets, R-PEARL, B-PEARL, SPE, and SignNet all use 4-layer GINE base models. Both R-PEARL and B-PEARL use a 3-layer GIN for Φ. To process positional encodings, in addition to a 3-layer GIN, SPE uses 16 3-layer MLPs on the Scaffold and Size splits, while SignNet uses a 3-layer MLP across all splits. At each layer of the base model, all models concatenate the original positional encodings with the input features. In the Assay and Size splits we use R-PEARL with $K = 14$ and 80 samples, while for the Scaffold split, we use $K = 16$ and 200 samples.

For the RelBench tasks, both R-PEARL and B-PEARL models use $K = 7$. The R-PEARLmodel employs a 5-layer GIN with 40 hidden units and 120 samples. The B-PEARLmodel uses either a 5-layer or a 7-layer GIN, depending on the task: 5 layers for post-post-related and 7 layers for user-post-comment, both with the same number of hidden units. For the SignNet models, we use an 8-layer GIN with batch normalization to generate positional encodings. The positional encodings from the models are incorporated as additional node features for each node. The original node features are generated by a Tabular ResNet model, which learns representations over the various node features. These combined features are then fed into the base GNN model. All models follow

the same setup as in RelBench for the base model, which employs a 2-layer ID-GNN. Training is conducted with a batch size of 20.

Table 5: Estimated runtime per epoch in Hours:Minutes for different models on RelBench.

Task	No PE	SignNet-largest	SignNet-smallest	B-PEARL	R-PEARL
user-post-comment	00:03	00:07	01:22	00:13	00:17
post-post-related	00:01	00:08	00:26	00:05	00:01

Table 5 presents the runtime of our end-to-end PE models on the RelBench tasks. Notably, our R-PEARLand B-PEARLachieve shorter runtimes compared to SignNet-smallest across both tasks.

For our experiments and model training pipeline we follow the codebases of (Huang et al.) and (Lim et al.), using Python, PyTorch Paszke et al. (2019), and the PyTorch Geometric Fey & Lenssen (2019) libraries. Our code can be found here: `https://github.com/codelakepapers/RPE-Framework`.

J Additional Experiments

J.1 Experiments on Graph Isomorphism

We conduct experiments on the Circular Skip Link (CSL) dataset (Murphy et al., 2019) which is the golden standard when it comes to benchmarking GNNs for graph isomorphism (Dwivedi et al., 2020). CSL contains 150 4-regular graphs, where the edges form a cycle and contain skip-links between nodes. Each graph consists of 41 nodes and 164 edges and belongs to one of 10 classes. Message-passing GNNs with WL-related PEs fail to classify these graphs and classification is completely random. This is due to the inability of the WL algorithm to handle regular graphs.

The proposed PEARL architectures, however, have no issue in processing regular graphs and achieve 100% classification accuracy. In particular, let Φ be a two-layer GNN, where each layer is defined by Eq. (3) with $K = 5$, $F_0 = F_1 = 1$, and $\sigma(\cdot) = \texttt{ReLU}(\cdot)$. The generated node PE $\boldsymbol{P}$ is processed by a summation graph pooling function to produce a scalar embedding for each graph. Then, both B-PEARL and R-PEARL can perfectly classify the CSL graphs with 100% classification accuracy, for any randomly generated trainable weights. This means that B-PEARL and R-PEARL can perfectly classify the CSL graphs without any training. For example let Φ consist of two identical layers with parameters $(h_0, h_1, h_2, h_3, h_4) = (0, 1, -\frac{1}{2}, \frac{1}{3}, -\frac{1}{4})$. The output $\mathbf{1}^T\boldsymbol{P}$ of B-PEARL is presented in Table 6. The output remains the same for all graphs within the same class, and differs distinctly for graphs belonging to different classes. Consequently, perfect classification accuracy can be achieved by feeding the B-PEARL encoding into a simple linear classifier or even a linear assignment algorithm.

Table 6: B-PEARL PE for every class of the CSL graphs. B-PEARL can perfectly classify the CSL graphs with 100% classification accuracy

	Class								
	0	1	2	3	4	5	6	7	8	9
0	27351.6	8800.2	25779.9	20458.4	17197.2	15861.3	24055.6	4106.8	17667.0

K Ablation Studies

K.1 Ablation on K

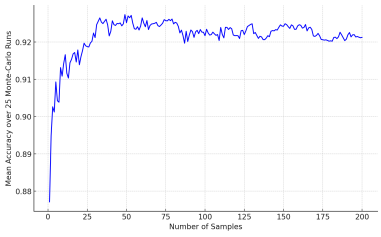
We also report our results on the ZINC dataset (Irwin et al., 2012) with alternate values of K for R-PEARL and B-PEARL . In the case of $K = 2$ we omit the first layer described in Eq. (3), using solely an 8-layer GIN for ϕ. These results are shown in Table7. We observe that even with $K = 2$ our model outperforms SignNet. Furthermore, even with a low K value of 4, B-PEARL outperforms SPE.

Table 7: logP Prediction in ZINC with different R-PEARL K values

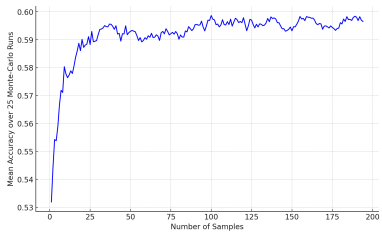
PE Method	#PEs	K	Test MAE	Training MAE	General. Gap
SignNet-8S	8	N/A	0.1034 ± 0.0056	0.0418 ± 0.0101	0.0602 ± 0.0112
SignNet	Full	N/A	0.0853 ± 0.0026	0.0349 ± 0.0078	0.0502 ± 0.0103
BasisNet-8S	8	N/A	0.1554 ± 0.0048	0.0513 ± 0.0053	0.1042 ± 0.0063
BasisNet	Full	N/A	0.1555 ± 0.0124	0.0684 ± 0.0202	0.0989 ± 0.0258
SPE-8S	8	N/A	0.0736 ± 0.0007	0.0324 ± 0.0058	0.0413 ± 0.0057
SPE	Full	N/A	0.0693 ± 0.0040	0.0334 ± 0.0054	0.0359 ± 0.0087
R-PEARL(ours)	N/A	1	0.0699 ± 0.002	0.0366 ± 0.006	0.0333 ± 0.007
R-PEARL(ours)	N/A	2	0.0831 ± 0.005	0.0725 ± 0.0125	0.0106 ± 0.008
R-PEARL(ours)	N/A	12	0.0699 ± 0.002	0.0366 ± 0.006	0.0333 ± 0.007
B-PEARL(ours)	N/A	1	0.0644 ± 0.001	0.0290 ± 0.003	0.0353 ± 0.002
B-PEARL(ours)	N/A	4	0.0680 ± 0.0023	0.0381 ± 0.004	0.0299 ± 0.0033
B-PEARL(ours)	N/A	12	0.0676 ± 0.0016	0.0403 ± 0.0104	0.0273 ± 0.0090

K.2 Ablation on number of samples M

We conduct ablation studies on the number of samples used by R-PEARL across the REDDIT datasets to examine the impact of sample size on model performance. These results are illustrated in Fig. 2. For each dataset, we evaluate the original model with 200 samples on a single test fold, varying the number of samples from 1 to 200. The test accuracy is then plotted, and Monte Carlo simulation-based smoothing is applied to generate the plots. Notably, we observe that model performance begins to converge with as few as 10 samples—an order of magnitude lower than the graph size.



(a) Sample Ablation on REDDIT-BINARY



(b) Sample Ablation on REDDIT-MULTI-5K

Figure 2: Ablation studies on the sample size for R-PEARL; It converges with only a few samples.

K.3 Ablation on the number of GNN layers in PEARL

Table 8: Ablation on the number of GIN layers in B-PEARL for the ZINC Dataset over 4 seeds.

# GIN Layers	3	5	7	9
Test MAE	0.0701 ± 0.005	0.067 ± 0.003	0.069 ± 0.001	0.0644 ± 0.001

The results in Table 8 show that B-PEARL achieves strong performance on the ZINC dataset even with less layers in the GNN producing the positional encodings. With only 5 layers, B-PEARL outperforms SPE.

K.4 ABLATION ON DIFFERENT PARAMETER SIZE

Table 9: logP Prediction in ZINC over number of parameters.

PE Method	#Parameters	Test MAE	Training MAE	General. Gap
SignNet	487k	0.0853 ± 0.0026	0.0349 ± 0.0078	0.0502 ± 0.0103
SPE	650k	0.0693 ± 0.0040	0.0334 ± 0.0054	0.0359 ± 0.0087
R-PEARL	644k	0.0699 ± 0.002	0.0366 ± 0.006	0.0333 ± 0.007
B-PEARL	644k	0.0644 ± 0.001	0.0290 ± 0.003	0.0353 ± 0.002
R-PEARL	487k	0.0721 ± 0.0045	0.0399 ± 0.001	0.0306 ± 0.0015
B-PEARL	487k	0.0679 ± 0.0026	0.0336 ± 0.008	0.0322 ± 0.004

Table 9 presents our ablation study on the number of parameters for the ZINC dataset. Notably, all models outperform SignNet even within the 500k parameter budget. Furthermore, B-PEARL achieves superior performance compared to SPE on full eigenvectors, even with fewer parameters.