Spectral Basis Learning for Expressive Graph Neural Networks in Link Prediction

Niloofar Azizi*[1], Nils M. Kriege[2,3], Nicholas J. A. Harvey[4], Horst Bischof[5]

[1]Cambridge Center for AI in Medicine, University of Cambridge, Cambridge, United Kingdom
[2]Faculty of Computer Science, University of Vienna, Vienna, Austria
[3]Research Network Data Science, University of Vienna, Vienna, Austria
[4]Department of Computer Science, University of British Columbia, Vancouver, BC, Canada
[5]Institute of Computer Graphics and Vision, Graz University of Technology, Graz, Austria

Abstract

Graph Neural Networks (GNNs) excel in handling graph-structured data but often underperform in link prediction tasks compared to classical methods, mainly due to the limitations of the commonly used message-passing principle. Notably, their ability to distinguish non-isomorphic graphs is limited by the 1-dimensional Weisfeiler-Lehman test (WL). Our study presents a novel method to enhance the expressivity of GNNs by embedding induced subgraphs into the eigenbasis of the graph Laplacian. We introduce a **L**earnable **L**anczos algorithm with **L**inear **C**onstraints (LLwLC), proposing two novel subgraph extraction strategies: encoding vertex-deleted subgraphs and applying Neumann eigenvalue constraints. For the former, we demonstrate the ability to distinguish graphs that are indistinguishable by 2-WL, while maintaining efficiency. The latter focuses on link representations enabling differentiation between k-regular graphs and node automorphism, a vital aspect for link prediction tasks. Our approach results in a lightweight architecture, reducing the need for extensive training datasets. Empirically, our method improves performance in challenging link prediction tasks across benchmark datasets, establishing its practical utility and supporting our theoretical findings. Notably, LLwLC achieves 20x and 10x speedups by requiring only 5% and 10% of the data from the PubMed and OGBL-Vessel datasets, while comparing to the state-of-the-art.

Code —
https://github.com/NiloofarAzizi/aaai-2026-LLwLC
Extended version — https://arxiv.org/abs/2408.12334

Introduction

Graphs play a crucial role in various domains, representing linked data such as social networks (Adamic and Adar 2003), citation networks (Shibata, Kajikawa, and Sakata 2012), knowledge graphs (Nickel et al. 2015), metabolic network reconstruction (Oyetunde et al. 2017), and user-item graphs in recommender systems (Monti, Bronstein, and Bresson 2017). Graph Neural Networks (GNNs) have emerged as state-of-the-art tools for processing graph-structured data. Message Passing Neural Networks (MPNNs) are the most prevalent technique within GNNs, which, relying on neighborhood aggregation, exhibit expressiveness no greater than the first-order Weisfeiler-Leman (1-WL) test (Weisfeiler and Leman 1968; Xu et al. 2019; Morris et al. 2019, 2023). Therefore, MPNNs cannot distinguish specific graph structures, *e.g.,* k-regular graphs. Moreover, link prediction (LP) tasks cannot always be answered reliably based on pairs of node embeddings obtained from MPNNs. Specifically, the node automorphism problem arises in instances where two nodes possess identical local structures, resulting in equal embeddings and, consequently, identical predictions for both when paired with the same third node. However, their relationships to this node, *e.g.,* in terms of distance, may differ (Zhang et al. 2021; Chamberlain et al. 2023).

Efforts aimed at augmenting the expressiveness of MPNNs have pursued four main directions: aligning with the k-WL hierarchy (Morris et al. 2019; Maron et al. 2019b; Azizian and Lelarge 2021), enriching node features with identifiers, exploiting structural information that cannot be captured by the WL test (Bodnar et al. 2021b,a), and Subgraph GNNs (SGNNs) (Bevilacqua et al. 2022; Guerra et al. 2022). SGNNs are a recent class of expressive GNNs that model graphs through a collection of subgraphs, extracted explicitly or implicitly. Subgraph extraction can be achieved, *e.g.,* by removing or marking specific nodes or directly counting specific substructures (Papp and Wattenhofer 2022). However, in the worst case, previous SGNNs involve computationally intensive preprocessing steps or running a GNN many times.

To address these limitations, we propose a novel approach grounded in graph signal processing (Ortega et al. 2018) and spectral graph theory (Chung and Graham 1997). Specifically, we introduce a new eigenbasis called **L**earnable **L**anczos with **L**inear **C**onstraints (LLwLC), which explicitly incorporates linear constraints—particularly those derived from induced subgraphs—into the basis. Our method builds on a low-rank approximation (Eckart and Young 1936) of the Laplacian matrix, utilizing a constrained variant of the Lanczos algorithm (Golub, Zhang, and Zha 2000). Similar to methods such as BUDDY (Chamberlain et al. 2023), LLwLC operates in two-hop neighborhoods. By learning expressive features, it achieves state-of-the-art performance with less training data, significantly reducing training time.

The LLwLC eigenbasis enhances feature expressiveness by incorporating linear constraints derived from the graph

*Work done while at Graz University of Technology.

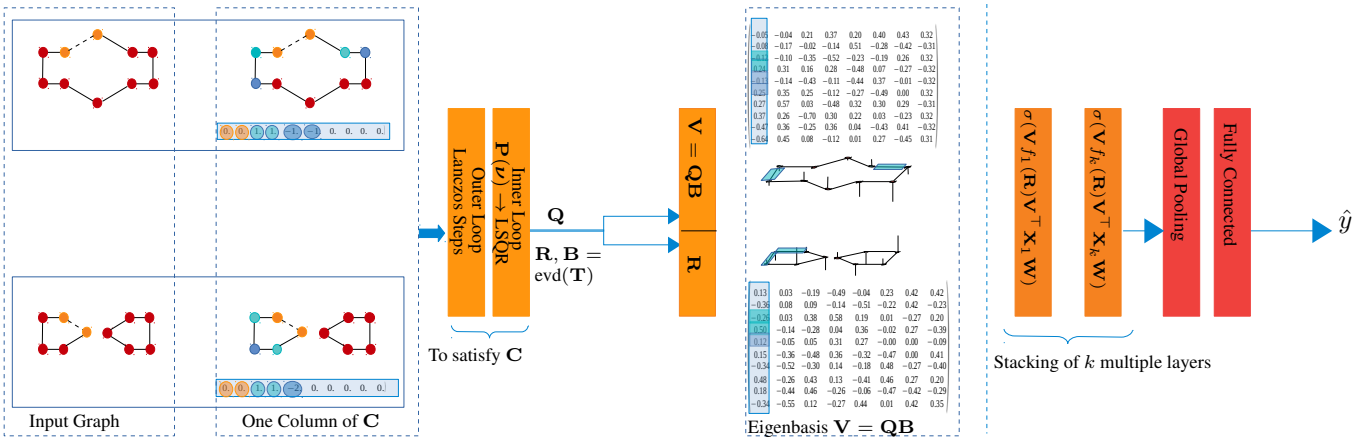

Figure 1: The comprehensive workflow of LLwLC: We introduce a novel eigenbasis to boost the expressivity of GNNs by introducing constraints. For the two 2-regular input graphs (first dotted block), the constraint matrix $\mathbf{C}$ (two columns shown in the second dotted block) is derived. Specifically, as depicted, for the first Neumann eigenvalue constraint (focused on link representation), we integrate the degrees of nodes into one column of the constraint matrix $\mathbf{C}$ such that $\mathbf{C}^\top \boldsymbol{f} = 0$ ensures $\boldsymbol{f}(x) - \boldsymbol{f}(y) = 0$ for adjacent nodes x and y in the boundary. To predict the dotted link, nodes up to two hops away are analyzed: one-hop in light blue and two-hop in dark blue, centered on yellow query nodes. We address an eigenvalue problem based on the input graph Laplacian matrix $\mathbf{L}$ under the linear constraints encoded by the matrix $\mathbf{C}$ using the Lanczos algorithm with the linear constraint consisting of an outer loop (Lanczos algorithm) and an inner loop ($\mathbf{P}(\nu)$ to $\mathcal{N}(\mathbf{C})$ with LSQR algorithm) resulting in the eigenpairs $\mathbf{V} = \mathbf{QB}$ and $\mathbf{R}$. The third block demonstrates constructing eigenbases for the two 2-regular graphs, ensuring features conform to constraints (indicated in blue). We stack multiple blocks; each block i computes $\mathbf{X}_{i+1} = \sigma(\mathbf{V} f_i(\mathbf{R}) \mathbf{V}^\top \mathbf{X}_i \mathbf{W})$, where f_i is an MLP on the eigenvalue matrix $\mathbf{R}$. The sequence ends with global pooling and a fully connected layer predicting link probability $\hat{y}$.

structure. We propose two novel subgraph extraction policies, focusing on vertex-deleted subgraphs and Neumann eigenvalue constraints. By leveraging vertex-deleted subgraphs, LLwLC can distinguish between graphs that are not separable using the 2-WL test, while Neumann constraints enable the encoding of boundary conditions and link representations between nodes, allowing differentiation of regular graphs that are indistinguishable by the WL test. Intuitively, our Neumann constraint represents the trapping of spectral energy by enforcing zero variation at its boundaries. This enables the grouping of graph edges at the boundary and facilitates the capture of fine-grained patterns such as cycles. Furthermore, the vertex-deleted subgraph constraint encodes global properties while remaining sensitive to vertex deletions, enhancing expressivity. Theoretical analysis indicates that LLwLC can be applied in various problem settings. We evaluate its effectiveness in link prediction tasks, where expressivity and the ability to distinguish automorphic nodes are pivotal.

Our research presents several key contributions: (i) We develop a novel, efficient eigenbasis (LLwLC) for encoding linear constraints, including induced subgraphs, using the Lanczos algorithm with linear constraints. (ii) We explore the application of Neumann eigenvalue constraints within this eigenbasis, which encodes induced subgraphs and link representations. (iii) We provide a comprehensive theoretical analysis of LLwLC's convergence. (iv) We investigate the impact of vertex-deleted subgraphs on improving the expressiveness and scalability of LLwLC. (v) We evaluate LLwLC on challenging link prediction tasks.

Preliminaries

Notations An undirected graph $G = (V, E, \mathbf{X})$ consists of a node (or vertex) set V, edge set E, and node features $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n is the number of nodes. Each row $\boldsymbol{x}_v \in \mathbb{R}^d$ denotes node v's features. $\mathbf{A}$ and $\mathbf{D}$ are the graph's adjacency and degree matrices, respectively. The graph Laplacian is $\mathbf{L} = \mathbf{D} - \mathbf{A}$ and we denote by $\mathbf{U}$ and $\mathbf{\Lambda}$ its eigenvector and eigenvalue matrices. The degree of a node v is d_v.

Spectral Graph Convolutional Networks For a graph signal $\boldsymbol{x} \in \mathbb{R}^n$, Shuman et al. (2013) define the graph Fourier transform, $\mathbf{U}^\top \boldsymbol{x}$, and its inverse, $\mathbf{U}\boldsymbol{x}$, based on the eigenbasis of the graph Laplacian $\mathbf{L}$. The graph convolution is $\mathbf{U}(\mathbf{U}^\top \boldsymbol{x} * \mathbf{U}^\top \boldsymbol{y}) = \mathbf{U} g(\mathbf{\Lambda}) \mathbf{U}^\top \boldsymbol{x}$ where $\boldsymbol{y}$ is the graph filter, g is the function applied over the eigenvalue matrix $\mathbf{\Lambda}$ to encode the graph filter, and $*$ is the elementwise multiplication. The seminal spectral GCN method (Bruna et al. 2013) is cubic in the number of nodes. To address this, different g functions were defined (Henaff, Bruna, and LeCun 2015; Azizi et al. 2022). LanczosNet (Liao et al. 2019) uses the Lanczos algorithm for fast multi-scale computation with learnable spectral filters. However, like prior GNNs, LanczosNet has limited expressivity and cannot distinguish regular graphs. To address this, we introduce a learnable spectral basis that encodes subgraphs as linear constraints.

Lanczos with Linear Constraint Networks

This section outlines the Lanczos Algorithm with Linear Constraints and the construction of a constraint matrix $\mathbf{C}$

for subgraph embedding, followed by the development of the complete LLwLC block and LLwLCNet pipeline. It also includes a proof of convergence properties and explores subgraph extraction policies, focusing on Neumann eigenvalue constraints and vertex-deleted subgraphs.

Lanczos Algorithm with Linear Constraints

For a given symmetric matrix $\mathbf{L} \in \mathbb{R}^{n\times n}$ and a randomly initialized vector $\boldsymbol{\nu} \in \mathbb{R}^n$, the κ-step Lanczos algorithm (Lanczos 1950) computes an orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{n\times m}$ and a symmetric tridiagonal matrix $\mathbf{T} \in \mathbb{R}^{m\times m}$, such that $\mathbf{Q}^\top \mathbf{L}\mathbf{Q} = \mathbf{T}$. We represent $\mathbf{Q}_N = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_N]$ where the column vector $\boldsymbol{q}_i$ corresponds to the i^{th} Lanczos vector. The matrices $\mathbf{B} \in \mathbb{R}^{m\times m}$ and $\mathbf{R} \in \mathbb{R}^{m\times m}$ represent the eigenvectors and eigenvalues of $\mathbf{T}$, respectively. By investigating the j^{th} column of the system $\mathbf{LQ} = \mathbf{QT}$ and rearranging terms, we obtain $\mathbf{L}\boldsymbol{q}_j = \beta_{j+1}\boldsymbol{q}_{j+1} + \beta_j \boldsymbol{q}_{j-1} + \alpha_j \boldsymbol{q}_j$.

Having the linear constraint changes the plain Lanczos algorithm by replacing $\boldsymbol{u}_j = \mathbf{L}\boldsymbol{q}_j - \beta_j \boldsymbol{q}_{j-1}$ with $\boldsymbol{u}_j = \boldsymbol{p}_j - \beta_j \boldsymbol{q}_{j-1}$ assuming the initial vector $\boldsymbol{\nu}$ is projected into the null space of the constraints (Golub, Zhang, and Zha 2000). Please note that the orthogonal projector $\mathbf{P}$ can be obtained through the QR decomposition of $\mathbf{C}$ when dealing with a dense constraint matrix $\mathbf{C}$. In situations where $\mathbf{C}$ is sparse and $\dim(\mathcal{N}(\mathbf{C}^\top)) \approx n$, the projector is given by $\mathbf{P} = \mathbf{I} - \mathbf{C}\mathbf{C}^\dagger$, with $\mathbf{C}^\dagger$ being the Moore-Penrose inverse of $\mathbf{C}$. Assuming $\mathbf{C}$ has full column rank, $\mathbf{C}^\dagger$ can be computed as $(\mathbf{C}^\top\mathbf{C})^{-1}\mathbf{C}^\top$ (Björck 1996). If we project the initial vector $\boldsymbol{\nu}$ into null space of the constraint matrix $\boldsymbol{\nu}_1 = \mathbf{P}\boldsymbol{\nu} \in \mathcal{N}(\mathbf{C}^\top)$ and notice the mathematical equivalence between computing the smallest eigenvalue of the constraint $A_p = \mathbf{P}^\top\mathbf{L}\mathbf{P}$ and $\mathbf{L}$ then one step of the Lanczos algorithm with the linear constraints is $\beta_{j+1}\boldsymbol{q}_{j+1} = \mathbf{P}\mathbf{L}\mathbf{P}\boldsymbol{q}_j - \beta_j\mathbf{P}\boldsymbol{q}_{j-1} - \alpha_j\mathbf{P}\boldsymbol{q}_j = \mathbf{P}(\mathbf{L}\boldsymbol{q}_j - \beta_j\boldsymbol{q}_{j-1} - \alpha_j\boldsymbol{q}_j)$. Algorithm 1 describes the steps of the Lanczos Algorithm with the Linear Constraints in detail.

This algorithm is structured into two main components: the 'outer loop', which is a straightforward Lanczos algorithm iteration, and the 'inner loop', which focuses on resolving the least squares problem expressed as $\mathbf{P}(\boldsymbol{b}) = \min_{y\in\mathbb{R}^l} \|\mathbf{C}\boldsymbol{y} - \boldsymbol{b}\|_2$. Here, $\boldsymbol{y}$ is defined as $\mathbf{C}^\dagger\boldsymbol{b}$, and $\boldsymbol{b}$ is $\mathbf{L}\boldsymbol{q}_j$.

LLwLCNet

In this part, we detail our approach to computing the eigenvectors of the graph Laplacian, ensuring they adhere to input graph constraints. We construct our eigenbasis by addressing a large, sparse, symmetric eigenvalue problem with homogeneous linear constraints. It requires minimizing

$$\min_{\mathbf{C}^\top\boldsymbol{f}=0, \boldsymbol{f}\neq 0} \frac{\boldsymbol{f}^\top\mathbf{L}\boldsymbol{f}}{\boldsymbol{f}^\top\boldsymbol{f}}, \tag{1}$$

where $\mathbf{C} \in \mathbb{R}^{n\times l}$ with $n \gg l$ is the constraint matrix.

To address the problem outlined in Equation 1, we utilize the Lanczos algorithm with the linear constraints, detailed in Algorithm 1. However, differing from the iterative approach for the least square equation suggested in the Lanczos algorithm with the linear constraints to solve $\mathbf{C}\boldsymbol{y} = \boldsymbol{b}$ we utilize the PyTorch framework (Paszke et al. 2017) for our computations. This choice is due to our sparse and not overly large constraint matrix allowing for the direct QR factorization (Anderson, Bai, and Dongarra 1992) within PyTorch, offering numerical stability and the capability for backpropagation. In the following, we describe constructing the constraint matrix $\mathbf{C}$, where we extract subgraphs first and derive the constraint matrix accordingly.

Algorithm 1: Lanczos Algorithm with Linear Constraint (LLwLC)

INPUT: $\mathbf{L}$, $\mathbf{P} = \mathbf{I} - \mathbf{C}(\mathbf{C}^\top\mathbf{C})^{-1}\mathbf{C}^\top$, $\boldsymbol{\nu}$, steps κ, tolerance ϵ
INIT: $\boldsymbol{\nu}_1 = \mathbf{P}(\boldsymbol{\nu})$, $\beta_1 = \|\boldsymbol{\nu}_1\|_2$, $\boldsymbol{q}_0 = 0$
for $j = 1$ **to** κ **do**
  $\boldsymbol{q}_j = \boldsymbol{\nu}_j / \beta_j$
  $\boldsymbol{p}_j = \mathbf{P}(\mathbf{L}\boldsymbol{q}_j)$
  $\boldsymbol{u}_j = \boldsymbol{p}_j - \beta_j \boldsymbol{q}_{j-1}$
  $\alpha_j = \boldsymbol{u}_j^\top \boldsymbol{q}_j$
  $\boldsymbol{\nu}_{j+1} = \boldsymbol{u}_j - \alpha_j \boldsymbol{q}_j$
  $\beta_{j+1} = \|\boldsymbol{\nu}_{j+1}\|_2$
  if $\beta_{j+1} \leq \epsilon$ **then**
    BREAK
  end if
end for
$Q = [q_1, \ldots, q_\kappa]$, construct $\mathbf{T}$
$\text{EVD}(\mathbf{T}) = \mathbf{B}\mathbf{R}\mathbf{B}^\top$
RETURN: $\mathbf{V} = Q\mathbf{B}$, $\mathbf{R}$

Algorithm 2: LLwLCNet: Spectral Network Using LLwLC Basis

INPUT: LLwLC output $\mathbf{V}, \mathbf{R}$, signal $\mathbf{X}$
INIT: $\mathbf{X}_0 = \mathbf{X}$
for $i = 0$ **to** $k - 1$ **do**
  $\mathbf{X}_{i+1} = \sigma(\mathbf{V} f_i(\mathbf{R})\mathbf{V}^\top \mathbf{X}_i \mathbf{W}_i)$
end for
$\hat{y} = \text{FC}(\text{GlobalPool}(\mathbf{X}_k))$
RETURN: $\hat{y}$

Constraint Matrix C The matrix $\mathbf{C}$ allows us to specify linear constraints that the graph Laplacian matrix's eigenvectors must satisfy. Each column of the constraint matrix represents a distinct constraint. We introduce two approaches to define the constraint matrix $\mathbf{C}$ on the basis of specific subgraphs: vertex-deleted subgraphs and Neumann eigenvalue constraints. A detailed explanation of how each column of the constraint matrix is built for these methods can be found in the next section.

Addressing the eigenvalue problem with linear constraints yields a tridiagonal matrix, denoted as $\mathbf{T}$, and an orthogonal matrix $\mathbf{Q}$. The decomposition of matrix $\mathbf{T}$ produces matrices $\mathbf{R}$ and $\mathbf{B}$. Here, $\mathbf{R}$ represents the Ritz eigenvalues, and $\mathbf{V} = \mathbf{QB}$ forms the eigenbasis that satisfies the constraints imposed by matrix $\mathbf{C}$. Forcing the eigenvectors to satisfy appropriately defined constraints leads to having different eigenbasis for graphs where the MPNN returns the same features. This is exemplified in the case described in Figure 1, where two 2-regular graphs yield two different eigenbases.

Full Block After determining the eigenbasis, we are ready to establish the full block of the Lanczos Layer with Lin-

ear Constraint (LLwLC). In this new eigenbasis, we develop spectral filters by applying a multilayer perceptron f to the eigenvalue matrix $\mathbf{R}$. With these learned filters, we reconstruct our basis and transform the graph signals $\mathbf{X} \in \mathbb{R}^{m \times n}$ into this basis to extract features that meet our specific constraints. $\hat{\mathbf{L}}$ is the graph Laplacian matrix computed from the low-rank approximation of the constrained eigenvalue problem. Each LLwLCNet block is

$$\sigma(\mathbf{V} f(\mathbf{R}) \mathbf{V}^\top \mathbf{X}^{(i)} \mathbf{W}) = \sigma(\hat{\mathbf{L}} \mathbf{X}^{(i)} \mathbf{W}). \quad (2)$$

Here, $\mathbf{W} \in \mathbb{R}^{n \times m}$ is the learnable weight matrix and σ is a non-linearity (ReLU in our case).

Full Architecture As represented in Algorithm 2, we increase the number of blocks to deepen our architecture and capture more complex features. Each block reuses the initially computed eigenbasis and applies a multi-layer perceptron (MLP) to the eigenvalue matrix $\mathbf{R}$ to reconstruct its corresponding $\hat{\mathbf{L}}$. Our complete pipeline concludes with a global sort pooling (Zhang et al. 2018) and a fully connected block in the last layer used to predict link existence, see Figure 1.

Subgraph Extraction Policy

A subgraph selection policy is a function $\pi \colon \mathcal{G} \to \mathbb{P}(\mathcal{G})$ assigning to a graph a subset of its subgraphs (Bevilacqua et al. 2022). Here, $\mathcal{G}$ is the set of all graphs with at most n nodes and $\mathbb{P}(\mathcal{G})$ is its power set. Although any linear constraint in the input graph satisfying the full rank assumption can be encoded in $\mathbf{C}$, we propose the following policies.

Neumann Eigenvalue The Neumann eigenvalue (Chung and Graham 1997) is $\lambda_S = \inf_{\boldsymbol{f} \neq 0} \frac{\sum_{x \in S} \boldsymbol{f}(x) \mathbf{L} \boldsymbol{f}(x)}{\sum_{x \in S} \boldsymbol{f}^2(x) d_x}$, subject to $\sum_{y \in S,\, x \in \delta S,\, y \sim x} (\boldsymbol{f}(x) - \boldsymbol{f}(y)) = 0$ and $\sum_{x \in S} \boldsymbol{f}(x) d_x = 0$. The function $\boldsymbol{f} \colon S \cup \delta S \to \mathbb{R}$ represents the Neumann eigenvector satisfying the Neumann conditions. The vertex boundary, δS, of an induced subgraph consists of all vertices not in S but adjacent to at least one vertex in S. Specifically, the first constraint encodes the link representation. Building on previous link prediction research, we consider nodes that are two hops away from the query nodes, where S represents the one-hop-away nodes, and δS denotes the boundary nodes between one-hop and two-hop-away nodes. It ensures that the aggregate of eigenvector differences across nodes equates to zero, as expressed by $\sum (\boldsymbol{f}(x) - \boldsymbol{f}(y)) = 0$. The column corresponding to the subgraph is constructed based on the equation. The degrees of the nodes involved in the constraint are entered into the column to satisfy the equation. Specifically, the degrees of nodes two hops away are negated, and the degrees of nodes one hop away are included only if they are connected to nodes two hops away. Entries for uninvolved nodes are set to 0.

Example 1. Consider the graph  , where the query nodes are orange, the nodes in S are blue, and the boundary nodes δS are green. The boundary constraint $\sum_{y \in S, x \in \delta S, y \sim x} (\boldsymbol{f}(x) - \boldsymbol{f}(y)) = 0$ is realized by the column $[0, 0, 1, 1, -1, -1, 0, 0, 0, 0]^\top$ of the matrix $\mathbf{C}$.

For induced subgraphs, we populate its corresponding column in $\mathbf{C}$ with the degrees of each node (and fill the remaining entries of the column with 0 for nodes not involved in the subgraph), ensuring that the derived eigenvectors fulfill the condition $\sum \boldsymbol{f}(x) d_x = 0$ (see next paragraph for details). This condition ensures that the features learned using this eigenbasis reflect the imposed constraints.

Thus, the Neumann eigenvalue problem can be reformulated as Equation 1 where $\mathbf{L} \in \mathbb{R}^{n \times n}$ is a symmetric and large sparse matrix, and $\mathbf{C} \in \mathbb{R}^{n \times l}$ (with $n \gg l$) is also large, sparse, and of full column rank. The time complexity involved in extracting subgraphs depends on the product of the maximum degree of nodes and the count of nodes in the boundary. When we enforce that the eigenvectors satisfy the constraints related to induced subgraphs and link representation, we ensure that the corresponding features adhere to these constraints.

Proposition 1. *Applying Neumann eigenvalue constraints to the eigenbasis (i) results in features with greater expressivity than MPNNs (or 1-WL), (ii) addresses the node automorphism problem, (iii) enables the distinction of specific non-isomorphic k-regular graphs (Proof in extended version).*

Vertex-deleted Subgraphs The second subgraph extraction policy we propose is based on vertex-deleted subgraphs. Given that we can encode any subgraph into our eigenbasis, we can examine whether a specific substructure collection can completely characterize each graph. By the reconstruction conjecture (Ulam 1960), we assume we can reconstruct the graph if we have all the $n - 1$ vertex-deleted subgraphs. According to Koval and Kwan (2023), exponentially many graphs are determined by their spectrum. Thus, we propose that each column of the constraint matrix corresponds to a vertex-deleted subgraph, and we include the degrees of the nodes within the corresponding subgraph in each column, representing deleted nodes by zero entries.

Example 2. Consider the graph  , where the red nodes are deleted, and the orange nodes are contained in the subgraph. The constraint $\sum \boldsymbol{f}(x) d_x = 0$ is represented by the column $[2, 2, 2, 2, 0, 0, 0, 0, 0, 0]^\top$ in the matrix $\mathbf{C}$.

We build the other columns of the constraint matrix accordingly. The intuition behind defining such a constraint is that it implicitly represents the vertex deleted subgraph. Learning the filters over the eigenvalue matrix of the graph Laplacian leads to defining an affinity matrix based on Gaussian functions and driving the corresponding adjacency matrix based on the threshold (Liao et al. 2019). Thus, $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$ where $\mathbf{W}$ is the affinity matrix (Coifman and Lafon 2006). $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ is the Markov transition matrix and its powers $\mathbf{P}^t$ are the probability of t-hop-away distance between the nodes i to node j. Thus, the diffusion map is defined based on the right eigenvector and eigenvalues of the Markov transition matrix $\mathbf{P}$. If λ_i and $\boldsymbol{\psi}_i$ are the eigenpairs of the matrix $\mathbf{P}$ then the eigenbasis of diffusion map after t-steps for the i^{th} value is $\boldsymbol{\Phi}_t(i) = (\lambda_1^t \psi_1(i), \lambda_2^t \psi_2(i), ...)$. Projecting to null space of the constraint matrix, $(\mathbf{I} - (\mathbf{C}^\top (\mathbf{C}\mathbf{C}^\top)^{-1} \mathbf{C}))$ cancels out the nodes with zero entries and considers the others based on the degree we put in the constraint matrix. The diffusion map embedding adapts to reflect diffusion patterns that are aware of the vertex deletions.

In extracting vertex-deleted subgraphs from a dense graph, the worst-case time complexity is $\mathcal{O}(n^3)$ due to the high number of edges in dense graphs, leading to increased computational load. For sparse graphs with fewer edges, the complexity is reduced by a factor of $\mathcal{O}(n)$. Our empirical results show that applying a few constraints significantly improves performance, making the practical overhead align linearly with the number of nodes. This approach is similar to the marker-based method (Papp and Wattenhofer 2022), as both consider slightly perturbed input graphs.

Utilizing vertex-deleted subgraphs allows us to distinguish graphs that the 2-WL method cannot distinguish. For instance, although the 4×4 rook and Shirkhande graphs are indistinguishable with 2-WL, their corresponding constraint matrices—constructed from the vertex-deleted subgraphs—lead to distinct eigenbases and diffusion behavior.

Stochastic Constraints C We discovered empirically that effective outcomes can be attained with a limited number of constraints, such as Neumann constraints, or in another ablation study involving only ten vertex-deleted subgraphs. This observation aligns with the findings of Bollobás (1990), who demonstrated that almost all graphs can be reconstructed using only three vertex-deleted subgraphs. Given the prohibitive cost of considering all n vertex-deleted subgraphs, our approach, inspired by Bevilacqua et al. (2022), involves selecting a subset of constraints. At each epoch, we stochastically choose k vertex-deleted subgraphs for analysis.

Lanczos Algorithm with Linear Constraint Convergence

In this section, we substantiate LLwLC eigenbasis's convergence properties by conducting an error analysis on perturbations and referencing Greenbaum's findings to demonstrate the existence of an exact Lanczos algorithm for any perturbed version. By establishing the upper bound for the Lanczos algorithm's low-rank approximation, we affirm the convergence of our LLwLC eigenbasis.

Perturbation and Error Study The accuracy of the linear least square problem using QR factorization depends on the precision of the QR factorization. As discussed by Zhang, Baharlouei, and Wu (2020), two types of accuracy errors are crucial in QR factorization when solving linear least square problems: The backward error for a matrix $\mathbf{Z}$ is defined as $\frac{\|\mathbf{Z}-\hat{\mathbf{Q}}\hat{\mathbf{R}}\|}{\|\mathbf{Z}\|}$ and the orthogonality error of $\hat{\mathbf{Q}}$ is measured by $\|\mathbf{I}-\hat{\mathbf{Q}}^\top\hat{\mathbf{Q}}\|$. Ideally, both numerical errors should be zero, but due to roundoff errors and the potential loss of orthogonality in the Gram-Schmidt QR process, the QR factorization might not be sufficiently accurate for solving the linear least square problem.

After examining the impact on accuracy, we analyze the theoretical gap between the exact Lanczos algorithm and its perturbed variant due to inexact QR factorization. The inexact QR factorization applied in the 'inner loop' will impact the accuracy of both the Lanczos vectors and the tridiagonal matrices produced. Consequently, the computed tridiagonal matrix $\mathbf{T}_j$ is a perturbed version of the theoretical tridiagonal matrix, denoted as $\mathbf{T}_j^*$, that would be generated by an exact Lanczos iteration. This relationship can be expressed as $\mathbf{T}_j = \mathbf{T}_j^* + \mathbf{E}_j$, where $\mathbf{E}_j$ is the perturbation matrix after the j^{th} step. The following theorem details the error bounds of the perturbed tridiagonal matrix in comparison to the exact solution of $\mathbf{T}$ after the j^{th} step of the Lanczos algorithm.

Theorem 1. *Let $\mathcal{U}$ and $\tilde{\mathcal{U}}$ be the eigenspaces corresponding to the smallest eigenvalues λ and $\tilde{\lambda}$ of the symmetric matrices $\mathbf{L}$ and $\tilde{\mathbf{L}} = \mathbf{L} + \mathbf{E}$, respectively. Then for any $\boldsymbol{u} \in \mathcal{U}$ and $\tilde{\boldsymbol{u}} \in \tilde{\mathcal{U}}$ with $\|\boldsymbol{u}\|_2 = 1$ and $\|\tilde{\boldsymbol{u}}\|_2 = 1$, we have*

$$\tilde{\lambda} - \lambda \approx \sum_{i=1}^{j} \mathbf{E}_j(i,i)\boldsymbol{u}(i)^2 + 2\sum_{i=1}^{j-1} \mathbf{E}_j(i,i+1)\boldsymbol{u}(i)\boldsymbol{u}(i+1),$$

where $\mathbf{E}_j(s,t)$ is the (s,t) element of $\mathbf{E}_j$.

After exploring the theoretical gap between exact and perturbed Lanczos algorithms, we investigate Greenbaum's result, which shows that each perturbed Lanczos corresponds to an exact version.

Greenbaum's Results (Greenbaum 1989) The tridiagonal matrix $\mathbf{T}_j$ generated at the end of the j^{th} *finite precision* Lanczos process satisfying $\mathbf{L}\mathbf{Q}_j = \mathbf{Q}_j\mathbf{T}_j + \beta_{j+1}\boldsymbol{q}_{j+1}\boldsymbol{e}^\top{}_j + \mathbf{F}_j$, where $\boldsymbol{e}_j^\top$ is a vector with the j^{th} component one and all the other components zero, $\mathbf{F} = (\boldsymbol{f}_1, \ldots, \boldsymbol{f}_j)$ is the perturbation term with $\|\boldsymbol{f}_j\|_2 \leq \epsilon\|\mathbf{L}\|_2, \epsilon \ll 1$, is the same as that generated by an exact Lanczos process but with a different matrix $\tilde{\mathbf{L}}$. The matrices $\mathbf{L}$ and $\tilde{\mathbf{L}}$ are close in the sense that for any eigenvalue $\lambda(\tilde{\mathbf{L}})$ of $\tilde{\mathbf{L}}$, there is an eigenvalue $\lambda(\mathbf{L})$ of $\mathbf{L}$ such that $|\lambda(\tilde{\mathbf{L}}) - \lambda(\mathbf{L})| \leq \|\mathbf{F}_j\|_2$. Therefore, in our case with the constant accuracy of the QR factorization, we can show $\mathbf{P}\mathbf{L}\mathbf{P}\tilde{\mathbf{Q}}_j = \tilde{\mathbf{Q}}_j\mathbf{T}_j + \beta_j\tilde{\boldsymbol{q}}_{j+1}\boldsymbol{e}_j^\top + \tilde{\mathbf{F}}_j$, where $\tilde{\mathbf{F}}_j = \mathbf{O}(\eta)$ with η corresponds to the accuracy of the QR method.

Having established each perturbed Lanczos algorithm corresponds to an exact Lanczos algorithm, we demonstrate the theorem below to bound the approximation error, as discussed in (Liao et al. 2019).

Theorem 2. *Let $\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ be the eigendecomposition of an $n \times n$ symmetric matrix $\mathbf{L}$ with $\mathbf{\Lambda}_{i,i} = \lambda_i, \lambda_1 \geq \cdots \geq \lambda_n$ and $\mathbf{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n]$. Let $\mathcal{U}_j \equiv span\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_j\}$. Assume κ-step Lanczos algorithm starts with vector $\boldsymbol{\nu}$ and outputs the orthogonal $\mathbf{Q} \in \mathbb{R}^{n\times\kappa}$ and tridiagonal matrix $\mathbf{T} \in \mathbb{R}^{\kappa\times\kappa}$. For any j with $1 < j < n$ and $\kappa > j$, we have*

$$\|\mathbf{L} - \mathbf{Q}\mathbf{T}\mathbf{Q}^\top\|_F^2 \leq \sum_{i=1}^{j} \lambda_i^2 \left(\frac{\sin(\boldsymbol{\nu},\mathcal{U}_i) \prod_{k=1}^{j-1} \frac{\lambda_k - \lambda_N}{\lambda_k - \lambda_j}}{\cos(\boldsymbol{\nu},\boldsymbol{u}_i) T_{\kappa-i}(1+2\gamma_i)} \right)^2 + \sum_{i=j+1}^{N} \lambda_i^2$$

where $T_{\kappa-i}(x)$ is the Chebyshev Polynomial of degree $\kappa - i$ and $\gamma_i = (\lambda_i - \lambda_{i+1})/(\lambda_{i+1} - \lambda_N)$.

Based on Greenbaum's results (Greenbaum 1989), for our computed perturbed Lanczos algorithm, exists an exact Lanczos algorithm but for a different matrix. Based on Theorem 2, we also cognize the upper bound of the low-rank approximator of the Lanczos algorithm. Thus, the perturbed Lanczos algorithm, caused by the inaccuracy of the QR method for solving the least square equation, converges to the upper bound of the low-rank approximation of the matrix of the exact Lanczos algorithm.

Time Complexity LLwLCNet's time complexity is $\mathcal{O}(\kappa E + k^2 n + k^3)$ for the outer loop (Lanczos algorithm),

	SEAL	BUDDY	NBFNet	LLwLC
Preprocessing	$\mathcal{O}(1)$	$\mathcal{O}(lE(d+h))$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Training (1 link)	$\mathcal{O}(Ed^2)$	$\mathcal{O}(l^2h+ld^2)$	$\mathcal{O}(Ed+nd^2)$	$\mathcal{O}(\kappa E+k^2n)$
Inference	$\mathcal{O}(Ed^2)$	$\mathcal{O}(l^2h+ld^2)$	$\mathcal{O}(Ed+nd^2)$	$\mathcal{O}(\kappa E+k^2n)$

(a) LP tasks: l: hops, h: BUDDY sketch, κ: eigenvectors, k: constraints.

	k-IGN	k-GNN	LLwLC	GSN (3-WL)
Preprocessing	-	-	$\mathcal{O}(n^3)$	$\mathcal{O}(n^k)$
Dense	$\mathcal{O}(n^k)$	$\mathcal{O}(n^k)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$
Sparse	$\mathcal{O}(n^k)$	$\mathcal{O}(n^k)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$

(b) Expressive models.

Table 1: Time complexity comparison.

the QR factorization, and computing the pseudo-inverse of $k \times k$ matrix, respectively, where $\kappa \ll n$ is the number of computed eigenvectors and $k \ll n$ is the number of linear constraints. Table 1a compares LLwLCNet with other LP methods. Aligned with (Barabási and Albert 1999) research and supported by our empirical experiments, it has been established that graph reconstruction can be effectively achieved with just a few constraints. This leads to time complexity linear to the number of nodes. Contrastingly, established k-WL expressive models, such as k-IGNs (Maron et al. 2019a) and k-GNNs (Morris et al. 2019), are known to have a higher time complexity of $\mathcal{O}(n^k)$.

Furthermore, we compare LLwLCNet's time complexity with GSN (Bouritsas et al. 2022), which improves expressivity by counting specific substructures. However, it relies on task-specific substructure selection to introduce a suitable inductive bias and deals with the subgraph isomorphism. In GSN, the preprocessing step, in general, is $\mathcal{O}(n^k)$ for a generic substructure of size k. In contrast, our experimental results demonstrate that LLwLCNet can capture graph properties without requiring task-specific prior knowledge. Table 1b provides an overview of the time complexities.

Experiments

We evaluate LLwLCNet on link prediction to assess its ability to handle node automorphism and leverage subgraphs. Comparisons are made against heuristics (CN, RA, AA), vanilla GNNs (GCN, SAGE), models modifying message-passing graphs (SEAL, NBFNet), and those using pairwise features (Neo-GNN, BUDDY, NCNC). Baseline results follow Chamberlain et al. (2023). We test on six benchmarks: Cora, Citeseer, Pubmed, OGBL-Collab, OGBL-Vessel, and OGBL-PPA; dataset details are in the extended version, and OGBL baselines follow the leaderboard.

Setup We train LLwLCNet using binary cross-entropy loss with a learning rate of 0.001 for 20 epochs. The model employs two 32-channel MLP layers with ReLU activations and 0.1 dropout per block, and caps the eigenpairs at 10 with zero padding. We implement the model in PyTorch (Paszke et al. 2017) and PyTorch Geometric (Fey and Lenssen 2019), and follow SEAL (Zhang and Chen 2018) with a 90/10 train–test split. For OGBL-PPA, we use 1-hop neighborhoods, vertex-deleted subgraph constraints, and four LLwLCNet blocks; other datasets use a single block.

Link Prediction Results While LLwLC is broadly applicable, we focus on link prediction (LP) to demonstrate its ability to handle node automorphism and substructure awareness, improving expressivity. Prior work (Chamberlain et al. 2023) shows two-hop neighborhoods suffices for LP.

As shown in Table 2, LLwLC performs strongly across benchmarks with few parameters. It outperforms prior methods on Planetoid with 0.02M parameters. On OGBL-Collab, LLwLC achieves SOTA with 0.02M, exceeds BUDDY (1.10M) and SEAL (0.50M), and reaches 67.50% HR@50 with 0.03M. On OGBL-Vessel, it matches strong baselines using 0.019M and 10% training data. On OGBL-PPA, LLwLC uses 0.06M parameters and 0.2% training data—25× less than 5%—while approaching the second-best, BUDDY (Chamberlain et al. 2023).

Ablation Studies We assess the impact of applying Neumann constraints $\mathbf{C}$ to the Laplacian $\mathbf{L}$ on three benchmarks (Table 3a), showing that $\mathbf{C}$ consistently improves results. Adding vertex-deleted subgraph constraints further boosts performance (Table 3b): with ten constraints, HR@50 on OGBL-Collab increases from 42.83 to 69.40, HR@100 on Cora from 90.80 to 93.10, with similar gains on PubMed. Constraint-count details are in the extended version.

Wall-time and Reduction Dataset Our study has shown an enhancement in model expressivity, leading to a decrease in the need for parameters. This results in less training data being required. As shown in Table 4, on the PubMed dataset, our model surpasses BUDDY (Chamberlain et al. 2023) and SEAL (Zhang and Chen 2018) using merel 10% of the training data, which results in a training speed that is 20 times faster than SEAL. Furthermore, it delivers performance on par with BUDDY while using just 1% of the training data within the same training duration and without requiring any preprocessing steps. This enhanced efficiency reflects a more efficient learning mechanism and significantly reduces training duration. Regarding the OGBL-Vessel dataset, our approach outperforms SEAL (Zhang and Chen 2018), demonstrating a 10-fold increase in training speed. It achieves similar outcomes at a speed 90 times faster using only 1% of the training data. These experiments were all carried out on a GeForce GT1030 GPU (CUDA 11.6, PyTorch 1.13).

Related Work

MPNN Expressivity GNN expressivity is often measured by their ability to distinguish non-isomorphic graphs. As no polynomial-time algorithm for solving the graph isomorphism problem is known, developing GNNs that are both expressive and efficient poses a major challenge. Xu et al. (2019) showed MPNNs expressivity is limited to 1-WL test. This limitation is crucial in real-world applications, as the 1-WL test cannot distinguish certain structures, *e.g.,* regular

Method	Cora HR@100	Citeseer HR@100	Pubmed HR@100	Collab HR@50	Vessel roc-auc	PPI HR@100
CN	33.92 ± 0.46	29.79 ± 0.90	23.13 ± 0.15	56.44 ± 0.00	48.49	27.65 ± 0.00
AA	39.85 ± 1.34	35.19 ± 1.33	27.38 ± 0.11	64.35 ± 0.00	48.49	32.45 ± 0.00
RA	41.07 ± 0.48	33.56 ± 0.17	27.03 ± 0.35	64.00 ± 0.00	n.a.	49.33 ± 0.00
GCN (Kipf and Welling 2017)	66.79 ± 1.65	67.08 ± 2.94	53.02 ± 1.39	44.75 ± 1.07	43.53	18.67 ± 1.32
SAGE (Hamilton, Ying, and Leskovec 2017)	55.02 ± 4.03	57.01 ± 3.74	39.66 ± 0.72	48.10 ± 0.81	49.89	16.55 ± 2.40
Neo-GNN (Yun et al. 2021)	80.42 ± 1.30	84.67 ± 2.16	73.93 ± 1.19	57.52	n.a.	49.13 ± 0.60
SEAL (Zhang and Chen 2018)	81.71 ± 1.30	83.89 ± 2.15	$75.54\pm$	$64.74\pm$	80.50	48.80 ± 3.16
NBFnet (Zhu et al. 2021)	71.65 ± 2.27	74.07 ± 1.75	58.73 ± 1.99	OOM	n.a.	OOM
Surel+ (Yin et al. 2023)	N.A.	N.A.	N.A.	64.10	85.73	N.A.
BUDDY (Chamberlain et al. 2023)	88.00 ± 0.44	92.93 ± 0.27	74.10 ± 0.78	65.94 ± 0.58	55.14	49.85 ± 0.20
NCNC (Wang, Yang, and Zhang 2024)	89.65 ± 1.36	93.47 ± 0.95	81.29 ± 0.95	66.61 ± 0.71	N.A.	61.42 ± 0.73
LLwLCNet	91.44 ± 0.50	93.4 ± 1.50	83.10 ± 0.50	66.86 ± 0.70	81.60 ± 1.50	40.38
# Params.	0.019M	0.018M	0.024M	0.026M	0.036M	0.038M

Table 2: Results on LP benchmarks with LLwLCNet (Neumann Constraints). Red and blue indicate best and second-best. LLwLCNet was trained on 10% of VESSEL and 0.2% of PPI.

Method	Cora (AUC)	Citeseer (AUC)	PubMed (AUC)
LanczosNet	94.5%	96.5%	97.2%
LLwLC (w. $\mathbf{L}$ & $\mathbf{C}$)	97.0%	98.1%	98.3%

(a) With/without Neumann constraints (using $\mathbf{L}$).

Method	Collab (HR@50)	Cora (HR@100)	PubMed (HR@100)
LanczosNet	42.58	90.80	77.18
LLwLCNet w. Neumann	66.86	91.44	83.10
LLwLCNet w. 10 Const.	69.40	93.10	82.28
# Params.	0.026M	0.019M	0.021M

(b) Performance under different constraint settings.

Table 3: Impact of subgraph constraints on performance.

Dataset		SEAL	BUDDY	GCN	LLwLC	LLwLC	LLwLC
PubMed	Pre-train (s)	0	5	0	0	0	0
	Train (s)	81	1	66	7	4	1
	Reduction	100	100	100	10	5	1
	Accuracy	75.54	74.10	53.02	80.15	78.20	70.84
	# Params.	0.486M	1.565M	0.052M	0.024M	0.024M	0.024M
OGBL-Vessel	Train (s)	12600	N.A.	12600	1200	700	140
	Reduction	100	100	100	10	5	1
	Accuracy	80.50	55.14	43.53	81.60	79.24	78.72
	# Params.	0.042M	N.A.	0.035M	0.036M	0.036M	0.036M

Table 4: Training time per epoch of LP models.

graphs, and does not capture several natural graph properties well, *e.g.,* distances and cycle counts (Li and Leskovec 2022). Recent studies have addressed these limitations with four approaches: adding random attributes to nodes (Sato, Yamada, and Kashima 2021), using deterministic positional features (Zhang and Chen 2018), developing higher-order GNNs to surpass the 1-WL test's expressivity limits (Maron et al. 2019b), and subgraph GNNs applying markings, such as the node-deletion approach of ESAN (Bevilacqua et al. 2022). See Morris et al. (2023) for a detailed survey. Our method follows the subgraph GNN line of research.

Subgraph GNNs for Link Prediction SEAL enhances WLNM (Zhang and Chen 2017), the first subgraph-based LP, by using graph convolutional layers and encoding positional features. SEAL demonstrates that information within two-hop subgraphs is sufficient, aligning with classical methods. Neo-GNN (Yun et al. 2021) and BUDDY decouple pairwise representation from node representation learning to reduce computational overhead but may oversimplify pairwise representations. Recent LP methods like NCNC (Wang, Yang, and Zhang 2024) and LPFormer (Shomer et al. 2024) combine GNNs with structural heuristics; in contrast, our approach jointly models subgraphs via a spectral eigenbasis, enabling richer interaction modeling with a lighter architecture.

Conclusion

In this work, we introduced LLwLC, a novel method designed to enhance the expressivity of GNNs. Through the incorporation of two novel subgraph extraction strategies, we constructed a lightweight architecture that minimizes reliance on extensive training datasets. Empirical results show that our method significantly improves performance in link prediction tasks across various benchmark datasets. Notably, the LLwLC achieved $20\times$ and $10\times$ speedups, requiring only 5% and 10% data from the PubMed and OGBL-Vessel datasets, respectively, compared to the state-of-the-art methods. These findings underscore the practical utility and theoretical advancement of our method, illustrating LLwLC's potential to exceed the expressivity of 2-WL and its ability to differentiate between k-regular graphs. A promising direction for future work is to explore the impact of learning linear constraints between nodes and edges within the input graph and encoding them in the eigenbasis of the Laplacian.

Acknowledgements

This research was funded by the Austrian Research Promotion Agency (FFG) under project no. 874065. Nils Kriege has been supported by the Vienna Science and Technology Fund (WWTF) and the City of Vienna [Grant ID: 10.47379/ICT22059]. The author sincerely appreciates the valuable feedback from Mohsen Fayyaz, Joshua Erde, and András Papp.

References

Adamic, L. A.; and Adar, E. 2003. Friends and Neighbors on the Web. *Social networks*.

Anderson, E.; Bai, Z.; and Dongarra, J. 1992. Generalized QR Factorization and its Applications. *Linear Algebra and its Applications*.

Azizi, N.; Possegger, H.; Rodolà, E.; and Bischof, H. 2022. 3D Human Pose Estimation using Möbius Graph Convolutional Networks. In *ECCV*.

Azizian, W.; and Lelarge, M. 2021. Expressive Power of Invariant and Equivariant Graph Neural Networks. In *ICLR*.

Barabási, A.-L.; and Albert, R. 1999. Emergence of Scaling in Random Networks. *science*.

Bevilacqua, B.; Frasca, F.; Lim, D.; Srinivasan, B.; Cai, C.; Balamurugan, G.; Bronstein, M. M.; and Maron, H. 2022. Equivariant Subgraph Aggregation Networks. In *ICLR*.

Björck, Å. 1996. *Numerical Methods for Least Squares Problems*. SIAM.

Bodnar, C.; Frasca, F.; Otter, N.; Wang, Y.; Lio, P.; Montufar, G. F.; and Bronstein, M. 2021a. Weisfeiler and Lehman Go Cellular: CW Networks. In *NeurIPS*.

Bodnar, C.; Frasca, F.; Wang, Y.; Otter, N.; Montufar, G. F.; Lio, P.; and Bronstein, M. 2021b. Weisfeiler and Lehman Go Topological: Message Passing Simplicial Networks. In *ICML*.

Bollobás, B. 1990. Almost Every Graph Has Reconstruction Number Three. *Journal of Graph Theory*.

Bouritsas, G.; Frasca, F.; Zafeiriou, S. P.; and Bronstein, M. 2022. Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting. *TPAMI*.

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral Networks and Locally Connected Networks on Graphs. In *ICLR*.

Chamberlain, B. P.; Shirobokov, S.; Rossi, E.; Frasca, F.; Markovich, T.; Hammerla, N. Y.; Bronstein, M. M.; and Hansmire, M. 2023. Graph Neural Networks for Link Prediction with Subgraph Sketching. In *ICLR*.

Chung, F. R.; and Graham, F. C. 1997. *Spectral Graph Theory*. American Mathematical Soc.

Coifman, R. R.; and Lafon, S. 2006. Diffusion Maps. *Applied and computational harmonic analysis*.

Eckart, C.; and Young, G. 1936. The Approximation of One Matrix by Another of Lower Rank. *Psychometrika*.

Fey, M.; and Lenssen, J. E. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLRW*.

Golub, G. H.; Zhang, Z.; and Zha, H. 2000. Large Sparse Symmetric Eigenvalue Problems with Homogeneous Linear Constraints: the Lanczos Process with Inner-Outer Iterations. *Linear Algebra and its Applications*.

Greenbaum, A. 1989. Behavior of Slightly Perturbed Lanczos and Conjugate-Gradient Recurrences. *Linear Algebra and its Applications*.

Guerra, M.; Spinelli, I.; Scardapane, S.; and Bianchi, F. M. 2022. Explainability in Subgraphs-enhanced Graph Neural Networks. *arXiv:2209.07926*.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*.

Henaff, M.; Bruna, J.; and LeCun, Y. 2015. Deep Convolutional Networks on Graph-structured Data. *arXiv:1506.05163*.

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

Koval, I.; and Kwan, M. 2023. Exponentially Many Graphs Are Determined by Their Spectrum. *arXiv:2309.09788*.

Lanczos, C. 1950. An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators. *J.Res.Natl.Bur.Stand.*

Li, P.; and Leskovec, J. 2022. The Expressive Power of Graph Neural Networks. *Graph Neural Networks: Foundations, Frontiers, and Applications*.

Liao, R.; Zhao, Z.; Urtasun, R.; and Zemel, R. S. 2019. Lanczosnet: Multi-scale Deep Graph Convolutional Networks. In *ICLR*.

Maron, H.; Ben-Hamu, H.; Serviansky, H.; and Lipman, Y. 2019a. Provably Powerful Graph Networks. In *NeurIPS*.

Maron, H.; Ben-Hamu, H.; Shamir, N.; and Lipman, Y. 2019b. Invariant and Equivariant Graph Networks. In *ICLR*.

Monti, F.; Bronstein, M.; and Bresson, X. 2017. Geometric Matrix Completion with Recurrent Multi-Graph Neural Networks. In *NeurIPS*.

Morris, C.; Lipman, Y.; Maron, H.; Rieck, B.; Kriege, N. M.; Grohe, M.; Fey, M.; and Borgwardt, K. 2023. Weisfeiler and Leman go Machine Learning: The Story so far. *Journal of Machine Learning Research*, 24(333): 1–59.

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2019. Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks. In *AAAI*.

Nickel, M.; Murphy, K.; Tresp, V.; and Gabrilovich, E. 2015. A Review of Relational Machine Learning for Knowledge Graphs. *IEEE*.

Ortega, A.; Frossard, P.; Kovačević, J.; Moura, J. M.; and Vandergheynst, P. 2018. Graph Signal Processing: Overview, Challenges, and Applications. *IEEE*.

Oyetunde, T.; Zhang, M.; Chen, Y.; Tang, Y.; and Lo, C. 2017. BoostGAPFILL: Improving the Fidelity of Metabolic Network Reconstructions Through Integrated Constraint and Pattern-based Methods. *Bioinformatics*.

Papp, P. A.; and Wattenhofer, R. 2022. A Theoretical Comparison of Graph Neural Network Extensions. In *ICML*.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic Differentiation in PyTorch. In *NeurIPS*.

Sato, R.; Yamada, M.; and Kashima, H. 2021. Random Features Strengthen Graph Neural Networks. In *SDM*.

Shibata, N.; Kajikawa, Y.; and Sakata, I. 2012. Link Prediction in Citation Networks. *JASIST*.

Shomer, H.; Ma, Y.; Mao, H.; Li, J.; Wu, B.; and Tang, J. 2024. LPFormer: An Adaptive Graph Transformer for Link Prediction. In *ACM SIGKDD*.

Shuman, D. I.; Narang, S. K.; Frossard, P.; Ortega, A.; and Vandergheynst, P. 2013. The Emerging Field of Signal Processing on Graphs: Extending High-dimensional Data Analysis to Networks and Other Irregular Domains. *IEEE*.

Ulam, S. M. 1960. *A Collection of Mathematical Problems*. Interscience Publishers.

Wang, X.; Yang, H.; and Zhang, M. 2024. Neural Common Neighbor with Completion for Link Prediction. In *ICLR*.

Weisfeiler, B.; and Leman, A. 1968. The Reduction of a Graph to Canonical Form and the Algebra which Appears Therein. *nti, Series*.

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

Yin, H.; Zhang, M.; Wang, J.; and Li, P. 2023. SUREL+: Moving from Walks to Sets for Scalable Subgraph-based Graph Representation Learning. In *VLDB*.

Yun, S.; Kim, S.; Lee, J.; Kang, J.; and Kim, H. J. 2021. Neo-gnns: Neighborhood Overlap-aware Graph Neural Networks for Link Prediction. In *NeurIPS*.

Zhang, M.; and Chen, Y. 2017. Weisfeiler-Lehman Neural Machine for Link Prediction. In *ACM SIGKDD*.

Zhang, M.; and Chen, Y. 2018. Link Prediction based on Graph Neural Networks. In *NeurIPS*.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An End-to-End Deep Learning Architecture for Graph Classification. In *AAAI*.

Zhang, M.; Li, P.; Xia, Y.; Wang, K.; and Jin, L. 2021. Labeling Trick: A Theory of Using Graph Neural Networks for Multi-Node Representation Learning. In *NeurIPS*.

Zhang, S.; Baharlouei, E.; and Wu, P. 2020. High Accuracy Matrix Computations on Neural Engines: A Study of QR Factorization and Its Applications. In *HPDC*.

Zhu, Z.; Zhang, Z.; Xhonneux, L.-P.; and Tang, J. 2021. Neural Bellman-ford Networks: A General Graph Neural Network Framework for Link Prediction. In *NeurIPS*.