Improving Spectral Graph Convolution for Learning Graph-level Representation

Mingqi Yang, Rui Li, Yanming Shen, Heng Qi, Baocai Yin

Abstract

From the original theoretically well-defined spectral graph convolution to the subsequent spatial based message-passing model, spatial locality (in vertex domain) acts as a fundamental principle of most graph neural networks (GNNs). In the spectral graph convolution, the filter is approximated by polynomials, where a $k$-order polynomial covers $k$-hop neighbors. In the message-passing, various definitions of neighbors used in aggregations are actually an extensive exploration of the spatial locality information. For learning node representations, the topological distance seems necessary since it characterizes the basic relations between nodes. However, for learning representations of the entire graphs, is it still necessary to hold? In this work, we show that such a principle is not necessary, it hinders most existing GNNs from efficiently encoding graph structures. By removing it, as well as the limitation of polynomial filters, the resulting new architecture significantly boosts performance on learning graph representations. We also study the effects of graph spectrum on signals and interpret various existing improvements as different spectrum smoothing techniques. It serves as a spatial understanding that quantitatively measures the effects of the spectrum to input signals in comparison to the well-known spectral understanding as high/low-pass filters. More importantly, it sheds the light on developing powerful graph representation models.

1 Introduction

The spatial localization (in vertex domain) rule can be traced back to the initial stage of graph neural network development, where researchers proposed to use the polynomial of (normalized) Laplacian to approximate desired filters to avoid eigendecomposition as well as the non-scalable problem (the amount of parameter of filters equals to the size of graphs) [24, 35, 63, 37, 85, 72, 14, 25, 34]. The locality is a side-effect of polynomials as the $k$-order polynomial covers the spatial $k$-hop substructure. It meets our empirical understanding of node-level prediction tasks since the topological distance provides the basic interaction information among nodes. Therefore, the locality is also considered as the evidence accounting for the effectiveness of polynomial filters and servers as the basic rule for almost all spectral methods [18, 42, 74]. Recently, [17] unifies various graph matrices used by GNNs, i.e. (normalized) Laplacian, adjacency matrix, etc, into Graph Shift Operators (GSOs) [60, 68, 21] and explores more effective GSO for GNNs. But GSO is also localized as the entry of the GSO matrix corresponding to the disconnected node pair is always fixed to 0. In addition to spectral graph convolution, the spatial locality rule further affects the subsequent spatial-based message-passing/neighborhood-aggregation models [22].

In message-passing, although there is no exact definition of the filter as that in spectral graph convolution, the rule of locality is preserved in aggregations, as most of them follow the pattern of the $k$-th layer covering $k$-hop local substructures [23, 75, 66, 16, 78]. A later layer in the deep model corresponds to a larger receptive field that can model long-range dependencies. In this perspective, GNN operation can be viewed as an extension of convolution neural networks (CNNs) [41, 26, 28] on non-Euclidean data. Analogously, there is extensive work leveraging the
power of deep GNNs to better capture long-range dependencies as well as complex structural patterns [76, 43, 49, 59, 31, 86, 87]. However, despite the common problems of deep neural networks such as exploding/vanishing gradients, deep GNNs further suffer from oversmoothing problem [44, 57]. Meanwhile, the existing convergence analysis with respect to the depth [45, 86] is under the restricted condition, i.e. connected, normalized, and the convergence results correspond to theoretical infinity depth, making them not well suitable for practical scenarios.

Complementary to developing deep architectures, some work proposed to explore neighbors beyond 1-hop in aggregations to model long-range dependencies, which can be viewed as a relaxation of the locality. Most of them are under the messaging-passing/neighborhood aggregation framework, where they empower the aggregation operation to better maintain the graph structural information by defining more complicated neighbors. For example, MixHop [1], DCNN [3] and Geom-GCN [58] aggregate neighbors at various distances to capture mixing relationships. k-GNN [59] uses high-order neighbors in aggregations to make it comparable to high-order Weisfeiler-Lehman graph isomorphism test [70] in distinguishing complex structures. Virtual node [22] adds a fully connected node to ensure that any node has connections with all others with at most distance 2. Graphormer [79] applies a fully connected structure and uses the shortest path distance matrix to characterize the graph structure. Interestingly, although there is no strict constraint of 1-hop neighbors, the definition of neighbors is always inspired by spatial information, which serves as the guidance to implementing various aggregations, including the popular Transformer architectures [65, 11, 30, 19, 82]. In summary, the spatial locality is an implicit property of filters in the spectral graph convolution, while in spatial methods, it explicitly guides the design of aggregations (from the vanilla 1-hop neighbors to various spatial neighbors in vertex domain).

For learning graph representations, an important research direction is expressiveness which measures the ability to encode complex structures [13, 47, 46, 6, 5]. The spatial-based methods, i.e. messaging-passing model is more popular than spectral-based methods since we can improve expressiveness by designing more sophisticated aggregations following the theory of Weisfeiler-Lehman graph isomorphism test [50, 75, 68, 61]. However, we found that the weak performance of spectral methods on learning graph representations is not caused by the weak expressiveness, and the explicit spatial information in the spatial methods is not necessary either. In contrast, that the spatial locality rule restricts to explore more powerful spectral filter is the reason for weak performance of spectral methods on learning graph representations. The localized polynomial basis requires sophisticated coefficients to maintain spectrum properties, making it ineffective to learn from scratch. Most existing spectral methods either apply the specially designed coefficients corresponding to high/low pass filters or learn the coefficients in the quite restricted case. By removing the locality constraint, we explore powerful filters with more flexibilities to better handle graph signals. As a result, the resulting model with no explicit spatial information involved can outperform existing messaging-passing models on learning graph representations. Our main contributions are:

1. We propose non-spatial graph convolution for graph-level representation learning. It breaks the spatial locality rule, and leverages the power of both spectral (more expressive graph filter) and spatial (more expressive aggregation operation) methods.

2. We quantify the effects of graph spectrum on input signals. It generalizes the oversmoothing analysis and is served as concrete evidence of the effectiveness of various spectrum smoothing operations as well as multi-head attention mechanisms and multi-aggregator methods. It further motivates the idea of channel-wise graph convolution to address the correlation issue.

3. We conduct extensive experiments to evaluate the necessity of the spatial locality as well as the effects of a more expressive filter for graph-level representation learning. We also compare our model with competitive baselines, showing its significant improvements.

## 2 Preliminaries

Let \( \mathcal{G} = (\mathcal{V}, \mathcal{E}) \) be an undirected graph with node set \( \mathcal{V} \) and edge set \( \mathcal{E} \). We denote \( N = |\mathcal{V}| \) the number of nodes, \( A \in \mathbb{A}^{N \times N} \) the adjacency matrix and \( H \in \mathbb{R}^{N \times d} \) the node feature matrix where \( d \) is feature dimensionality. \( h \in \mathbb{R}^N \) is a graph signal that corresponds to one dimension of \( H \).

**Spectral Graph Convolution** [24, 18]. The definition of spectral graph convolution relies on Fourier transform on the graph domain. For a signal \( h \) and graph Laplacian \( L = U \Lambda U^T \), we have Fourier
transform $\hat{x} = U^T x$, with inverse transform $x = U \hat{x}$. Then, the graph convolution of a signal $h$ with a filter $g_\theta$ is

$$g_\theta * h = U \left((U^T g_\theta) \odot (U^T h)\right) = U \hat{G}_\theta U^T h,$$

where $\hat{G}_\theta$ denotes a diagonal matrix in which the diagonal corresponds to spectral filter coefficients. To avoid eigendecomposition and ensure scalability, $\hat{G}_\theta$ is approximated by a truncated expansion in terms of Chebyshev polynomials $T_k(\tilde{\Lambda})$ up to the $k$-th order \cite{Defferrard2016}, which is also the polynomials of $\Lambda$,

$$\hat{G}_\theta(\Lambda) \approx \sum_{i=0}^{k} \theta_i T_i(\tilde{\Lambda}) = \sum_{i=0}^{k} \theta_i \Lambda^i,$$

where $\tilde{\Lambda} = \frac{2}{\Lambda_{\text{max}}} \Lambda - I_N$. Now the convolution in Eq. 1 is

$$U \hat{G}_\theta U^T h \approx U \left( \sum_{i=0}^{k} \theta_i \Lambda^i \right) U^T h = \sum_{i=0}^{k} \theta_i L^i h.$$  

(3)

Note that this expression is $k$-localized since it is a $k$-order polynomial in the Laplacian, i.e., it depends only on nodes that are at most $k$ steps away from the central node ($k$-hop neighbor).

**Graph Convolutional Network (GCN)** \cite{Kipf2017}. GCN is derived from 1-order Chebyshev polynomials with several approximations. They further introduce the renormalization trick $I_N + D^{-\frac{1}{2}} AD^{-\frac{1}{2}} \rightarrow \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}}$, with $\hat{A} = A + I_N$, $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$. Also, GCN can be generalized to multiple input channels and a layer-wise model:

$$H^{(l+1)} = \sigma \left( \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right),$$  

(4)

where $W$ is learnable matrix and $\sigma$ is nonlinear function.

**Graph Diffusion Convolution (GDC)** \cite{Ying2018}. A generalized graph diffusion is given by the diffusion matrix:

$$H = \sum_{k=0}^{\infty} \theta_k T^k,$$

(5)

with the weight coefficients $\theta_k$ and the generalized transition matrix $T$. $T$ can be $T_{rw} = AD^{-1}$, $T_{sym} = D^{-\frac{1}{2}} AD^{-\frac{1}{2}}$ or others as long as they are convergent. GDC can be viewed as a generalization of the original definition of spectral graph convolution, which also applies polynomial filters but not necessarily the Laplacian.

3 Non-spatial Graph Convolution

3.1 Spectral Transformation

Many improvements on GNNs correspond to a specific polynomial transformation where the resulting matrix shares the eigenspace with the original graph matrix, e.g., residual connection, initial residual connection \cite{Huang2019}, Personalized PageRank (PPR) \cite{Kou2009}, etc. Here, we discuss the properties of applying transformations on graph matrices in a more generalized perspective beyond polynomials, including the uniqueness with respect to the original graph and the relations to graph isomorphism. For a matrix $S$, we use $E_{(S, \lambda)}$ to denote the eigenspace of $S$ associated with $\lambda$ such that $E_{(S, \lambda)} = \{v : (S - \lambda I)v = 0\}$.

**Proposition 1.** Given a symmetric matrix $S \in \mathbb{R}^{n \times n}$ with $S = P \Lambda P^T$ where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, and $P$ can be any eigenbasis of $S$, let $\tilde{S} = P \phi(\Lambda) P^T$, where $\phi(\cdot)$ is an entry-wise function applied on $\Lambda$. Then we have

(i) $E_{(S, \lambda_i)} \equiv E_{(\tilde{S}, \phi(\lambda_i))}, \forall i \in [n]$;

(ii) Meanwhile, if $\phi(\cdot)$ is injective, $E_{(S, \lambda_i)} = E_{(\tilde{S}, \phi(\lambda_i))}$ and $\mathcal{F}_{\phi}(S) = P \phi(\Lambda) P^T$ is injective.

We prove Proposition 1 in Appendix A. Proposition 1 shows that the eigenspace of $\tilde{S}$ involves the eigenspace of $S$. Therefore, $\tilde{S}$ is invariant to the choice of eigenbasis, i.e., $\tilde{S} = P \phi(\Lambda) P^T = P' \phi(\Lambda) P'^T$ for any eigenbases $P$ and $P'$ of $S$. Hence, $\tilde{S}$ is unique to $S$ for a given $\phi(\cdot)$. Consistently,
we denote the mapping $F_\phi(S) = F_\phi(P\Lambda P^T) = P\phi(\Lambda)P^T$. We can implement the transformation $F_\phi(\cdot)$ with different $\phi(\cdot)$ to generate a series of matrix representations for a given graph. The injectivity of $F_\phi(\cdot)$ serves as a guarantee that the transformation results are distinguishable for different graphs as long as $\phi(\cdot)$ is injective.

General graph neural networks follow the basic principle that the learned representation is invariant to graph isomorphism which is also known as permutation invariance \[81\][52]. For any two graphs $G_1$ and $G_2$ with matrix representation $S_1$ and $S_2$ (e.g., adjacency matrix, Laplacian matrix, etc.), $G_1$ and $G_2$ are isomorphic if and only if there exists a permutation matrix $M$ such that $MS_1M^T = S_2$. We denote $I(S) = MSM^T$ and $f_{\text{GNN}}(S)$ the learned graph representation by a specific GNN model. Then $f_{\text{GNN}}(I(S)) = f_{\text{GNN}}(S)$.

**Claim 1.** $F_\phi(\cdot)$ is equivariant to graph isomorphism, i.e. $F_\phi(I(S)) = I(F_\phi(S))$.

**Proof.**

\[
F_\phi(I(S)) = F_\phi(MSM^T) = F_\phi(M(P\Lambda P^T)M^T) \\
= F_\phi((MP)\Lambda(MP)^T) = (MP)\phi(\Lambda)(MP)^T \\
= M(P\phi(\Lambda)P^T)M^T = I(F_\phi(S))
\]

As a result, $f_{\text{GNN}}(F_\phi(I(S))) = f_{\text{GNN}}(I(F_\phi(S))) = f_{\text{GNN}}(F_\phi(S))$. The permutation invariance is held by the transformation $F_\phi(\cdot)$.

$F_\phi(\cdot)$ unifies many existing transformations such as residual connection, i.e. $\phi(\lambda) = (1 - \alpha)\lambda + \alpha$, or PPR, i.e. $\phi^{(k+1)}(\lambda) = (1 - \alpha)\lambda^{(k)}(\lambda) + \alpha$. It also indicates a new direction of exploring $F_\phi(\cdot)$ beyond polynomials. The polynomial transformations in spectral methods are used to address the concern of eigendecomposition on large-scale graphs, while graphs in real-world graph-level predictions are much smaller than that in node-level predictions, making it viable to leverage sophisticated transformations to achieve improvements. For example, $k$-GNN \[50\] computes high-order neighbors introduced in Weisfeiler- Lehman graph isomorphism test to empower aggregations. Graphormer \[79\] computes the shortest path distance (SPD) matrix for each graph with $O(n^3)$ complexity to adapt to their fully connected Transformer architecture. DGN \[4\] adopts eigendecomposition to compute the first $k$ eigenvectors.

For most messaging-passing models, the aggregation operations are fully based on the understanding of the spatial information. In contrast, by applying the transformation $F_\phi(\cdot)$, we generate various matrix representations for a given graph without explicitly involving the spatial information, e.g., the degree of nodes, the distance of neighbors, etc. Proposition 1(ii) provides the theoretical support that we can conduct the learning process on the resulting matrix without information loss. In the following sections, we will explore more powerful graph convolution for learning graph representations leveraging the power of more sophisticated $F_\phi(\cdot)$.

### 3.2 Non-spatial Basis: $S^*$

Since graph signals are discrete, we reformulate the polynomial filter in Eq. 2 in the discrete case. We denote $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)^T \in \mathbb{R}^n$ and $\theta = (\theta_1, \theta_2, \ldots, \theta_k)^T \in \mathbb{R}^k$. $g_\theta$ is a collection of all filter coefficients, i.e., $\text{diag}(g_\theta) = \hat{G}_\theta(\Lambda)$. Then,

\[
g_\theta = \sum_{i=0}^{k} \theta_i \lambda^i = (\lambda^1, \lambda^2, \ldots, \lambda^k) \times \theta = V \times \theta, \quad (6)
\]

where $V \in \mathbb{R}^{n \times k}$ and $n$ is the number of nodes. If $\lambda_i \neq \lambda_j$ for any $i \neq j$, i.e., the algebraic multiplicity of all eigenvalues is 1, $V$ is a Vandermonde matrix with $\text{Rank}(V) = \min(n, k)$. Hence, a larger $k$ helps to better approximate the desired filter. When $k = n$, $V$ is a full-rank matrix and $g_\theta$ is able to represent any desired filter with proper assignments of $\theta$. Note that $n$ is much smaller in real-world graph-level tasks than that in node-level tasks, making $k = n$ more tractable.

To achieve more expressive filters, i.e. leverage the power of more basis, we define the transformation on the symmetric matrix $S$ based on the understanding in Section 3.1

\[
S^* = F_\phi(S) = P|\Lambda|^tP^T, \quad (7)
\]
where $\epsilon \in (0, 1)$. For each $|\lambda| \in \Lambda$, $|\lambda|^\epsilon = e^{\epsilon \ln |\lambda|}$. $S$ is a graph-related matrix such as Laplacian or adjacency matrix with no requirement on the bound of spectrum. Then, we replace the polynomial basis $L$ in Eq. 3 with $S^\epsilon$. The resulting graph filter becomes $G_\theta(|\lambda|^\epsilon) = \sum_{i=0}^k \theta_i |\lambda|^\epsilon i$. Correspondingly, the graph convolution of signal $\tilde{h}$ with the filter is

$$P \tilde{G}_\theta(|\lambda|^\epsilon) P^T \tilde{h} = P \sum_{i=0}^k \theta_i |\lambda|^\epsilon i P^T \tilde{h} = \sum_{i=0}^k \theta_i S^\epsilon i \tilde{h}. \quad (8)$$

Eq. 8 can leverage the power of more bases to approximate a more sophisticated filter while alleviating numerical instabilities by controlling $\epsilon \cdot k$ in a small range. Therefore, $S^\epsilon$ can be considered as a basis-augmentation technique as in Fig. 1, which does not have to require $S$ to be convergent. In contrast, general spectral methods require a bounded spectrum to ensure that $\lambda^k$ is convergent with respect to $k$, which is achieved by the (symmetry) normalization on $A$, i.e. $\bar{D}^{-\frac{1}{2}} \bar{A} \bar{D}^{-\frac{1}{2}}$, $\bar{D}^{-1} \bar{A}$ to guarantee its spectrum is bounded by $[-1, 1]$ or on $L$, i.e. $I - \bar{D}^{-\frac{1}{2}} \bar{A} \bar{D}^{-\frac{1}{2}}$ to ensure the boundary $[0, 2]$ and then rescale it to $[0, 1]$ to guarantee its spectrum is bounded by $[-1, 1]$ or on $L$, i.e. $I - \bar{D}^{-\frac{1}{2}} \bar{A} \bar{D}^{-\frac{1}{2}}$ to ensure the boundary $[0, 2]$ and then rescale it to $[0, 1]$ to ensure the boundary $[0, 2]$ and then rescale it to $[0, 1]$ to ensure the boundary $[0, 2]$ and then rescale it to $[0, 1]$ to ensure the boundary $[0, 2]$ and then rescale it to $[0, 1]$ to ensure the boundary $[0, 2]$ and then rescale it to $[0, 1]$. Although there is no concern of numerical instabilities of these methods when involving more basis, $\lambda^k$, especially for a small $\lambda$, inclines to 0 for a larger $k$. Also, for $S$ with unsmooth spectrum, a larger $k$ results in $S^k$ an ill-conditioned matrix which will be discussed in Section 3.3. All these problems make the higher-order basis ineffective in the practical limited precision condition. Meanwhile, from the spatial perspective, the requirements of normalizations among neighbors may lead to a restriction to explore more expressive aggregation/propagation schemes.

Also, general spectral methods follow the spatial locality rule, where a $k$-order polynomials always corresponds to spatial $k$-hop neighbors. A spatial $(k + 1)$-hop distance dependency within a graph would not be covered. Therefore, they require a larger $k$ or are combined with a multi-layer architecture to detect the possible long-range dependency. Interestingly, for $S^\epsilon$, there is no explicit spatial information as shown in Fig. 2. The choice of $k$ only relates to the expressiveness of the filter with no concern of the spatial receptive field.

We call the graph convolution in Eq. 7 non-spatial graph convolution. It decouples the spatial locality property from the filter, as a result, allowing to explore more powerful filter. However, also indicates the concern of overfitting occurred in more expressive filters in node-level learning tasks. They alleviate it by applying the simplest 1-order Chebyshev polynomials (an affine approximation). In experiments, we will investigate whether more expressive filters exacerbate overfitting for learning graph representations as well as the necessity of spatial locality.
3.3 Spectral Analysis of $S^\epsilon$

We have shown how we explore more powerful filter $S^\epsilon$ without the restriction of the spatial locality rule. Another benefit of selecting $S^\epsilon$ is that it better maintains the spectrum properties: when using $S^\epsilon$ as basis as in Eq. 8, it inherently results in a smoother spectrum compared with $S$.

The spectrum of the underlying graph matrix plays a crucial role in the spectral graph convolution. Various improvements on GNNs can be unified into the spectrum smoothing/shrinking operations \cite{12, 38}, some of which relate it to low pass filters \cite{71, 56, 36, 37} and some others relate it to the alleviation of the oversmoothing problem \cite{44, 57, 76, 45, 86}. In most of these methods, the spectrum properties are preserved by the polynomial coefficients. Although theoretically we can set $k = n$ and then obtain any desired filter $q_0$ by learning proper $\theta$ according to Eq. 6, in reality, it is less likely to learn $\theta$ in a complete free manner \cite{37, 27}. The polynomials with specially designed coefficients to explicitly modify spectrum, i.e. Personalized PageRank (PPR), heat kernel \cite{37}, etc or with the coefficients learned under the constrained condition, i.e. Chebyshev \cite{18}, Cayley \cite{42}, Bernstein \cite{27} polynomial, etc acts as the practical applicable filters. This is probably because the polynomial filter relies on sophisticated coefficients to maintain spectrum properties, which are less likely to be learned from scratch. However, by replacing the general basis, i.e. (normalized) adjacency/Laplacian matrix with $S^\epsilon$, $S^\epsilon$ inherently possesses a smoother spectrum compared with $S$, which relaxes the requirement on the coefficients in the perspective of maintaining the spectrum smoothness. Also, compared with $S^k$, $S^\epsilon$ is less likely to result in an ill-conditioned matrix when applying a larger $k$.

In addition to the existing spectral analysis, i.e. high/low-pass filters \cite{71, 56, 36, 37} (the signal processing perspective), oversmoothing phenomenon \cite{44, 57, 76, 45, 86} (the convergence perspective), etc, we study the effects of graph spectrum on the correlation among signals, which serves as a generalization of existing convergence analysis as well as the theoretical motivation to generalize the non-spatial graph convolution to the multi-channel scenario that will be discussed in Section 3.4. We denote $S = U \left( \sum_{i=0}^{k} \theta_i A^i \right) U^T$ in Eq. 8 for simplicity. Then $S = S^\epsilon$ and the convolution on $h$ is $Sh$. The cosine distance between $Sh$ and the $i$-th eigenvector $p_i$ of $S$ is

$$ \cos(\langle Sh, p_i \rangle) = \frac{h^T p_i \lambda_i}{\sqrt{\sum_{j=1}^{n} (h^T p_j)^2}} = \frac{\alpha_i \lambda_i}{\sqrt{\sum_{j=1}^{n} \alpha_j^2 \lambda_j^2}}. $$  \hspace{1cm} (9)

$\alpha_i = h^T p_i$ is the weight of $h$ on $p_i$, when representing $h$ with the group of orthonormal bases $p_i, i \in [n]$. Also, for any given two vectors $h$ and $h'$, we have

$$ \cos(\langle Sh, Sh' \rangle) = \frac{\sum_{i=1}^{n} \alpha_i \beta_i \lambda_i^2}{\sqrt{\sum_{i=1}^{n} \alpha_i^2 \lambda_i^2} \sqrt{\sum_{i=1}^{n} \beta_i^2 \lambda_i^2}}. $$  \hspace{1cm} (10)

where $\beta_i = h'^T p_i$. The detailed derivation of Eq. 9 and Eq. 10 are given in Appendix B.

Eq. 9 and Eq. 10 reveal that the cosine distance after convolution is only decided by the smoothness of $S$’s spectrum and the original direction of the signal. The graph signals tend to turn the direction to the eigenvectors corresponding to the larger eigenvalues and meanwhile, be orthogonal to the eigenvectors corresponding to the smaller eigenvalues, which means they will lose information in the direction of eigenvectors with eigenvalue 0. This problem is more critical in deep architectures:

**Proposition 2.** Assume $S \in \mathbb{R}^{n \times n}$ is a symmetric matrix with real-valued entries. $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$ are $n$ real eigenvalues, and $p_i \in \mathbb{R}^n, i \in [n]$ are corresponding eigenvectors. Then, for any given $h, h' \in \mathbb{R}^n$, we have

(i) $|\cos(\langle S^{k+1} h, p_1 \rangle)| \leq |\cos(\langle S^k h, p_1 \rangle)|$ and $|\cos(\langle S^{k+1} h, p_n \rangle)| \leq |\cos(\langle S^k h, p_n \rangle)|$ for $k = 0, 1, 2, \ldots, +\infty$;

(ii) If $|\lambda_1| > |\lambda_2|$, $\lim_{k \to +\infty} |\cos(\langle S^k h, p_1 \rangle)| = \lim_{k \to +\infty} |\cos(\langle S^k h, S^\epsilon h' \rangle)| = 1$, and the convergent speed is decided by $\frac{1}{|\lambda_2|}$.

We prove Proposition 2 in Appendix C. For general GNNs that generate the graph representations with the last hidden representations, a deeper architecture enlarges the receptive field but also violates the spectrum’s smoothness. As a result, the input signals would be more correlated to

\footnote{Here, nonlinearity is not involved in the propagation step. This meets the case of the decoupling structure where a multi-layer GNN is split into independent propagation and prediction steps \cite{45, 71, 36, 38, 84}.}
each other. Finally, \( \text{Rank}(\{h_1, h_2, \ldots, h_d\}) = 1 \), the information within signals would be washed out. Note that all the above analysis does not impose any constraint to the underlying graph such as connectivity. Also Proposition 2(i) further studies the monotonicity of convergence, which well differentiates it from existing convergence analysis.

**Comparisons with oversmoothing.** Our analysis generalizes the well-known oversmoothing phenomenon \([76, 45, 56]\). In our analysis, the convergence of the cosine distance among signals does not limit a graph to be connected or normalized that is required in the oversmoothing analysis analogical to the stationary distribution of the Markov chain, and even does not require a model to be necessarily deep: it is essentially caused by the bad distributions of eigenvalues, while the deep architecture exacerbates it. Interestingly, inspired by this perspective, the mentioned problem actually relates to the specific topologies since different topologies correspond to different spectrum. There exists topologies inherently possessing bad distributions of eigenvalues. They will suffer from the problem even with a shallow architecture. More importantly, Proposition 2(i) further takes the symmetry into consideration, showing that the convergence of cosine with respect to \( k \) is also monotonous. This provides more concrete evidence in the practical finite depth case that a deeper architecture could be more harmful than a shallow one, and can also be viewed as the explanation for why we can achieve improvements on the finite depth architectures to avoid problems occurring in the theoretical infinite case.

### 3.4 Multi-channel Generalizations

In Section 3.3, we study the effects of graph’s spectrum on the correlations among signals. It provides a new interpretability of the effectiveness of various spectrum shrinking operations as well as our proposed non-spatial basis. More interestingly, since the correlation problem occurs in the case of the shared filter, it naturally motivates us the strategy of each channel assigned an independent filter. Correspondingly, the convolution in Eq. 8 is generalized to multi-channel scenario as:

\[
\left( \sum_{i=0}^{k} \theta_{i,1} S^{e^{-i}} h_1, \sum_{i=0}^{k} \theta_{i,2} S^{e^{-i}} h_2, \ldots, \sum_{i=0}^{k} \theta_{i,d} S^{e^{-i}} h_d \right),
\]

where the node feature matrix \( H = (h_1, h_2, \ldots, h_d) \) and \( \theta \in \mathbb{R}^{k \times d} \). The entire architecture is given in Appendix D.

### 4 Related Work

**Spectrum smoothing techniques.** Various improvements on GNNs can be unified into the spectrum smoothing operations, some of which interpret it as low-pass filters \([63, 71, 56, 51, 88, 37]\), where low eigenvalues corresponding to large-scale structure in the graph (e.g. clusters \([54]\)) are amplified, while high eigenvalues corresponding to fine details but also noise are suppressed. In Section 3.3, we show a simple yet quantitative perspective different from existing analysis to accurately measure the effects of the graph spectrum on signals. It provides a new perspective to understand the effectiveness of various spectrum smoothness techniques: a smoother spectrum helps to alleviate the correlations among signals.

**Multi-head/aggregator.** From the original GAT \([66]\) to most recent various Transformer-based graph models \([2]\), multi-head attention works pretty well on graph tasks. Its effectiveness is mainly based on the understanding of the vanilla multi-head attention mechanism as that in various other tasks. Meanwhile, multi-aggregator methods \([16, 78]\) leverage the power of multiple aggregators to improve the discrimination of aggregations. Although multi-head and multi-aggregator are motivated by different insights and seem to have no connection with each other, according to Section 3.3 these methods all help to maintain the information among signals: each head/aggregator forms a propagation matrix \( S \) with its own spectrum. Multi-channel signals processed with different \( S \) are less likely to correlate to each other. Motivated by the understanding of the correlation problem, we directly assign each channel with an independent filter as Eq. 11 without additional subspace projection and concatenation operations as that in the multi-head attention or multi-aggregator methods.

---

\[^{7}\] propagation involving nonlinearity remains unexplored due to its high complexity, except for one case of ReLU as nonlinearity \([57]\). Most convergence analysis (such as over-smoothing) only study the simplified linear case \([12, 45, 71, 56, 86, 76, 49, 88, 37]\).
Learnable polynomial filters. ChebNet [18] approximates the filter with Chebyshev polynomials, and learns the coefficients of the Chebyshev basis via training. Similarly, BernNet [27] applies Bernstein polynomials with learnable coefficients of the Bernstein basis. GPR-GNN [15] directly learns the coefficients of a polynomial filter. Our method is orthogonal to all these existing methods that we propose non-spatial $S^\epsilon$ as the polynomial basis, which aims to improve spectral graph convolution on learning graph representations and can be easily integrated into others.

5 Experiments

We conduct experiments on TU [77, 33] and OGB [29] datasets which involve graph classification tasks and ZINC [20] which involves graph regression tasks. Then, we evaluate the effects of our proposed non-spatial basis and channel-independent graph convolution through elaborate ablation studies.

5.1 Results

Table 1: Results on TU dataset.

| dataset | NCI1 | NCI109 | ENZYMES | PTC_MR |
|---------|------|--------|----------|--------|
| GK [62] | 62.49±0.27 | 62.35±0.3 | 32.70±1.20 | 55.65±0.5 |
| RW [69] | NA | NA | 24.16±1.64 | 55.91±0.3 |
| PK [53] | 82.54±0.5 | NA | NA | 59.5±2.4 |
| FGSD [67] | 79.80 | 78.84 | NA | 62.8 |
| AWE [72] | NA | NA | 35.77±5.93 | NA |
| DGCNN [61] | 74.44±0.47 | NA | 51.0±7.29 | 58.59±2.5 |
| PSCN [55] | 74.44±0.5 | NA | NA | 62.29±5.7 |
| DCNN [13] | 56.61±1.04 | NA | NA | NA |
| ECC [84] | 76.82 | 75.03 | 45.67 | NA |
| DGK [77] | 80.31±0.46 | 80.32±0.3 | 53.43±0.91 | 60.08±2.6 |
| CapsGNN [73] | 78.35±1.55 | NA | 54.67±5.67 | NA |
| DiffPool [80] | NA | NA | 62.53 | NA |
| GIN [75] | 82.7±1.7 | NA | NA | 64.6±7.0 |
| k-GNN [50] | 76.2 | NA | NA | 60.9 |
| Non-spatial GN | 84.20±1.80 | 83.65±1.10 | 71.50±5.84 | 67.73±6.8 |

Settings. We use the default dataset splits for OGB and ZINC. For TU dataset, we follow the standard 10-fold cross-validation protocol and splits from [83] and report our results following the protocol described in [75, 80]. Following all baselines on the leaderboard of ZINC, we control the number of parameters around 500K. For non-spatial graph convolution, we set $\epsilon = \{0.3, 0.5\}$ and $k = \{6, 9\}$. Detailed hyperparameter settings are summarized in the supplementary file.

Results. Table 1, 2 and 3 summarize performance of Non-spatial GN comparing with baselines on TU, ZINC and MolPCBA datasets. For TU dataset, we report the results of each model in its original paper by default. When the results are not given in the original paper, we report the best testing results given in [83, 52, 73]. For ZINC and MolPCBA, we report the results of their public leaderboards and also involve recent proposed competitive models. TU involves small-scale datasets. NCI1 and NCI109 are around 4K graphs. ENZYMES and PTC_MR are under 1K graphs. General GNNs easily suffer from overfitting on these small-scale data, and therefore we can see that some traditional kernel-based methods even get better performance. However, non-spatial GN gets higher classification accuracies by a large margin on all these datasets. The results on TU show that although non-spatial GN achieves more expressive filters, it does not lead to overfitting on learning graph representations. Recently, Transformer-based models are quite popular in learning graph representations, and they significantly improve the results on large-scale molecular datasets. On ZINC, non-spatial GN outperforms these Transformer-based models by a large margin. And on MolPCBA, our non-spatial GN is also competitive compared with SOTA results. The Transformer-based methods leverage the power of multi-head attention mechanism. In non-spatial GN, it is implemented as the simple and straightforward channel-wise graph convolution based on the understanding of the correlation problem. The results show that this simplification does not result in any performance drop. Although our method and DGN [4] all require eigendecomposition in the preprocessing step, non-spatial GN significantly outperforms DGN on ZINC and MolPCBA.
Table 2: Results on ZINC.

| method               | #param. | test MAE  | method               | #param. | AP (%)   |
|----------------------|---------|-----------|----------------------|---------|----------|
| GIN [78]             | 509,549 | 0.526±0.051| GCN [35]            | 0.56M   | 20.20±0.24|
| GraphSage [23]       | 505,341 | 0.398±0.002| GIN [78]            | 1.92M   | 22.66±0.28|
| GAT [60]             | 531,345 | 0.384±0.007| GCN-VN [35]         | 2.02M   | 24.24±0.34|
| GCN [35]             | 505,079 | 0.367±0.011| GIN-VN [75]         | 3.37M   | 27.03±0.23|
| GatedGCN-PE [63]     | 505,011 | 0.214±0.006| GCN-VN+FLAG [38]    | 2.02M   | 24.83±0.37|
| MPNN (sum) [22]      | 480,805 | 0.145±0.007| GIN-VN+FLAG [38]    | 3.37M   | 28.34±0.38|
| PNA [16]             | 387,155 | 0.142±0.010| DeeperG-VN+FLAG [43]| 5.55M   | 28.42±0.43|
| DGN [4]              | -       | 0.168±0.003| PNA [16]            | 6.55M   | 28.38±0.35|
| GSN [7]              | 523,201 | 0.101±0.010| DGN [4]             | 6.73M   | 28.85±0.30|
| GT [19]              | 588,929 | 0.226±0.014| GINE-VN [10]        | 6.15M   | 29.17±0.15|
| SAN [39]             | 508,577 | 0.139±0.006| GINE-APNF [10]      | 6.15M   | 29.79±0.30|
| GraphormerFSLIM [79] | 489,321 | 0.122±0.006| PHC-GNN [40]        | 1.69M   | 29.47±0.26|

Non-spatial GN 500,961 0.0721±0.0028 Non-spatial GN 1.74M 29.25±0.32
Non-spatial GN 6.6M 0.0683±0.0039

5.2 Ablation Studies

We perform ablation studies on ZINC to investigate two components introduced in the non-spatial graph convolution: the non-spatial basis $S^\epsilon$ and the channel-wise filter (or the independent filter). All models are restricted to be 500K parameters like that in Table 2. We report the results of 5 runs with random seeds $\{0, 1, 2, 3, 4\}$.

Table 4: Properties of all tested bases.

|               | $\tilde{A}$ | $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ | $\tilde{D}^{-1}\tilde{A}$ | $\tilde{A}^\epsilon$ |
|---------------|-------------|-------------------------------------------------|---------------------------|------------------------|
| Bounded spectrum | ✓           | ✓                                              | ✓                         | ✓                      |
| Spatial localized       | ✓           | ✓                                              | ✓                         | ✓                      |

We empirically evaluated $\tilde{A}$, $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, $\tilde{D}^{-1}\tilde{A}$ and $\tilde{A}^\epsilon$ as polynomial basis with properties summarized in Table 4. $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ and $\tilde{D}^{-1}\tilde{A}$ are normalized ones and have bounded spectrum which are extensively studied by various spectral methods. $\tilde{A}$ does not involve normalization among neighbors which is usually applied by message-passing models.

![Figure 3: Ablation study results on ZINC with different settings.](image)

Does more basis gain improvements? In Figure 3, we systematically evaluate the effects of basis on learning graph representations, including $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, $\tilde{D}^{-1}\tilde{A}$ and our proposed $\tilde{A}^\epsilon$ with $\epsilon = 1/3, 1/6, 1/7, 1/8$. All implementations apply the proposed multi-channel generalization as in Eq. 11. $\tilde{A}$ is not involved in Figure 3 because it suffers from the numerical instability problem when $k > 5$ in our tests. Both $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ and $\tilde{D}^{-1}\tilde{A}$ gain improvements when applying a larger
We propose the non-spatial polynomial basis. It improves general spectral filters in learning graph representations. A
basis with the best test MAE close to 0.09 and the best valid MAE close to 0.11. In contrast, \( A' \) always
outperforms \( \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \) and \( \tilde{D}^{-1} \tilde{A} \) by a large margin for all different \( k \). And the best test MAE
converges to 0.07. The number of basis in \( A' \) is controlled by both \( \epsilon \) and \( k \). We use the tuple \((\epsilon, k)\) to
denote a combination of \( \epsilon \) and \( k \). By fixing the \( \epsilon \), the curves corresponding to \( \epsilon = 1/3 \) and \( \epsilon = 1/6 \)
show that increasing \( k \) gains improvements. By fixing the upper bound of \( \epsilon \times k \) to be 1, \((1/6, 6)\) involves 3 more basis than \((1/3, 3)\) and outperforms \((1/3, 3)\). The same results are also reflected in
the comparison of \((1/6, 12)\) and \((1/3, 6)\). For the comparison of \((1/6, 18)\) and \((1/3, 9)\), both settings
achieve the lowest MAE and the difference is less obvious.

**Stability.** Figure 3 also shows that the results of \( A' \) are more stable than \( \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \) and
\( \tilde{D}^{-1} \tilde{A} \) in different runs. For \( \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \) and \( \tilde{D}^{-1} \tilde{A} \), the difference between the best and the worst run can be
more than 0.02. While for \( A' \), this difference is less than 0.01. More results are given in Appendix E. The instablity is probably because learning \( \theta \) from scratch with no constraints is difficult to maintain
spectrum properties and therefore easily fall into an ill-posed filter [27]. The results show that \( A' \) inherently with smoother spectrum alleviates the problem and makes it more proper in the scenario of learning coefficients from scratch.

**Does deeper architecture gain improvements?** Some methods correspond to the weighted summation of the representations of all layers [49, 56, 89]. [37] unifies them to graph diffusion convolution. The depth is analagous to the number of basis. In this perspective, involving more basis is equivalent to stacking deeper architectures. Note that these methods use the power of \( \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \) as basis to ensure convergence when going deeper. And a larger power corresponds to the spatial larger receptive field. \( A' \) can be considered as a new strategy for developing deep graph neural networks with empirical promising results. The deep architecture of \( A' \) is motivated by the better approximation ability while having no explicit spatial information involved.

**Is spatial locality necessary for learning graph representation?** Different from existing methods, \( \tilde{A}' \) is obtained by directly applying transformations on the spectrum without explicitly considering spatial information (such as the number of neighbors for each node or the distance between a node pair). Interestingly, the actual performance does not reflect any drop, which serves as concrete evidence that explicit spatial information is not necessary for learning graph representation.

**Shared/independent channel.** On all 4 bases, the channel-wise filter (or the independent filter) outperforms the shared filter by a large margin. In the view of practical implementation, the channel-wise filter simply replaces the scalar-vector multiplication with the vector-vector dot product. It is much simpler than existing multi-head or multi-aggregator methods, but well handles the correlation problem with no performance compromise. We also found that the shared filter is more unstable in different runs (with a large std) than the channel-wise filter. This is probably because different channels may pose different patterns, which causes interference among each other in the case of the shared filter. While in the independent case, this problem is well avoided.

| Channel | Basis | test MAE | valid MAE |
|---------|-------|----------|-----------|
| shared | independent | \( A \) | \( \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \) | \( \tilde{D}^{-1} \tilde{A} \) | \( A' \) | 
| ✔️ | ✔️ | 0.1263±0.02606 | 0.1473±0.03280 |
| ✔️ | ✔️ | 0.1415±0.00748 | 0.1568±0.00729 |
| ✔️ | ✔️ | 0.1439±0.00900 | 0.1569±0.00739 |
| ✔️ | ✔️ | 0.1123±0.00836 | 0.1325±0.01131 |
| ✔️ | ✔️ | 0.1181±0.00654 | 0.1491±0.00365 |
| ✔️ | ✔️ | 0.0944±0.00379 | 0.1100±0.00787 |
| ✔️ | ✔️ | 0.0982±0.00417 | 0.1172±0.00666 |
| ✔️ | ✔️ | 0.0721±0.00282 | 0.0872±0.00256 |

6 Conclusion

We propose the non-spatial polynomial basis. It improves general spectral filters in learning graph representations and also serves as concrete evidence that spatial locality information is not necessary
for learning graph representations. Meanwhile, the consideration of cosine distance provides a quantitative perspective of the effects of graph spectrum on signals.

References

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayer Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In international conference on machine learning, pages 21–29. PMLR, 2019.

[2] Anonymous. SpecTRA: Spectral transformer for graph representation learning. In Submitted to The Tenth International Conference on Learning Representations, 2022. under review.

[3] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In Advances in Neural Information Processing Systems, pages 1993–2001, 2016.

[4] Dominique Beani, Saro Passaro, Vincent Létourneau, Will Hamilton, Gabriele Corso, and Pietro Liò. Directional graph networks. In International Conference on Machine Learning, pages 748–758. PMLR, 2021.

[5] Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yu Guang Wang, Pietro Liò, Guido Montúfar, and Michael Bronstein. Weisfeiler and lehman go cellular: Cw networks. In Advances in Neural Information Processing Systems, volume 34, 2021.

[6] Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F Montufar, Pietro Liò, and Michael Bronstein. Weisfeiler and lehman go topological: Message passing simplicial networks. In Marina Meila and Tong Zhang, editors, Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pages 1026–1037. PMLR, 18–24 Jul 2021.

[7] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. arXiv preprint arXiv:2006.09252, 2020.

[8] Xavier Bresson and Thomas Laurent. Residual gated graph convnets. arXiv preprint arXiv:1711.07553, 2017.

[9] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergeyyst. Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine, 34(4):18–42, 2017.

[10] Rémy Brossard, Oriel Frigo, and David Dehaene. Graph convolutions that can finally model local structure. arXiv preprint arXiv:2011.15069, 2020.

[11] Deng Cai and Wai Lam. Graph transformer for graph-to-sequence learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 7464–7471, 2020.

[12] Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie yan Liu, and Liwei Wang. Graphnorm: A principled approach to accelerating graph neural network training, 2020.

[13] Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gns. In Advances in Neural Information Processing Systems, pages 15868–15876, 2019.

[14] Xiuyuan Cheng, Zichen Miao, and Qiang Qiu. Graph convolution with low-rank learnable local filters. In International Conference on Learning Representations, 2021.

[15] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In International Conference on Learning Representations, 2021.

[16] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In Advances in Neural Information Processing Systems, 2020.
[17] George Dasoulas, Johannes F. Lutzeyer, and Michalis Vazirgiannis. Learning parametrised 
graph shift operators. In International Conference on Learning Representations, 2021.

[18] Michäel Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks 
on graphs with fast localized spectral filtering. In Advances in neural information processing 
systems, pages 3844–3852, 2016.

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

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

[21] Fernando Gama, Alejandro Ribeiro, and Joan Bruna. Stability of graph neural networks to 
relative perturbations. In ICASSP 2020-2020 IEEE International Conference on Acoustics, 
Speech and Signal Processing (ICASSP), pages 9070–9074. IEEE, 2020.

[22] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural 
message passing for quantum chemistry. In Proceedings of the 34th International Conference 
on Machine Learning-Volume 70, pages 1263–1272. JMLR. org, 2017.

[23] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large 
graphs. In Advances in Neural Information Processing Systems, pages 1024–1034, 2017.

[24] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral 
graph theory. Applied and Computational Harmonic Analysis, 30(2):129–150, 2011.

[25] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. The spectral graph wavelet 
transform: Fundamental theory and fast computation. In Vertex-Frequency Analysis of Graph 
Signals, pages 141–175. Springer, 2019.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image 
recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, 
pages 770–778, 2016.

[27] Mingguo He, Zhewei Wei, Zengfeng Huang, and Hongteng Xu. Bernnet: Learning arbitrary 
graph spectral filters via bernstein approximation. In NeurIPS, 2021.

[28] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep 
Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural 
networks for acoustic modeling in speech recognition: The shared views of four research groups. 
IEEE Signal processing magazine, 29(6):82–97, 2012.

[29] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele 
Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. 
arXiv preprint arXiv:2005.00687, 2020.

[30] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In 
Proceedings of The Web Conference 2020, pages 2704–2710, 2020.

[31] Wenbing Huang, Yu Rong, Tingyang Xu, Fuchun Sun, and Junzhou Huang. Tackling over-
smoothing for general graph convolutional networks. arXiv preprint arXiv:2008.09864, 2020.

[32] Sergey Ivanov and Evgeny Burnaev. Anonymous walk embeddings. In Jennifer Dy and Andreas 
Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 
80 of Proceedings of Machine Learning Research, pages 2191–2200, Stockholmsmässan, 
Stockholm Sweden, 10–15 Jul 2018. PMLR.

[33] Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. 
Benchmark data sets for graph kernels, 2016. http://graphkernels.cs.tu-dortmund.de

[34] Renata Khasanova and Pascal Frossard. Graph-based isometry invariant representation learning. 
In International conference on machine learning, pages 1847–1856. PMLR, 2017.
[35] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations (ICLR), 2017.

[36] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In International Conference on Learning Representations (ICLR), 2019.

[37] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. Advances in Neural Information Processing Systems, 32:13354–13366, 2019.

[38] Kezhi Kong, Guohao Li, Mucong Ding, Zuxuan Wu, Chen Zhu, Bernard Ghanem, Gavin Taylor, and Tom Goldstein. Flag: Adversarial data augmentation for graph neural networks. arXiv preprint arXiv:2010.09891, 2020.

[39] Devin Kreuzer, Dominique Beaini, William Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. arXiv preprint arXiv:2106.03893, 2021.

[40] Tuan Le, Marco Bertolini, Frank Noé, and Djork-Arné Clevert. Parameterized hypercomplex graph neural networks for graph classification. arXiv preprint arXiv:2103.16584, 2021.

[41] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[42] Ron Levie, Federico Monti, Xavier Bresson, and Michael M. Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. IEEE Transactions on Signal Processing, 67(1):97–109, 2019.

[43] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns. arXiv preprint arXiv:2006.07739, 2020.

[44] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[45] Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 338–348, 2020.

[46] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, pages 2156–2167, 2019.

[47] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. arXiv preprint arXiv:1812.09902, 2018.

[48] Gonzalo Mateos, Santiago Segarra, Antonio G Marques, and Alejandro Ribeiro. Connecting the dots: Identifying network structure via graph signal processing. IEEE Signal Processing Magazine, 36(3):16–43, 2019.

[49] Zhewei Wei Ming Chen, Bolin Ding Zengfeng Huang, and Yaliang Li. Simple and deep graph convolutional networks. 2020.

[50] 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 AAAI Conference on Artificial Intelligence, volume 33, pages 4602–4609, 2019.

[51] Balciilar Muhammet, Renton Guillaume, Héroux Pierre, Gaüzère Benoit, Adam Sébastien, and Paul Honeine. When spectral domain meets spatial domain in graph neural networks. In Thirty-seventh International Conference on Machine Learning (ICML 2020)-Workshop on Graph Representation Learning and Beyond (GRL+ 2020), 2020.
[52] Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. In International Conference on Learning Representations, 2019.

[53] Marion Neumann, Roman Garnett, Christian Bauckhage, and Kristian Kersting. Propagation kernels: efficient graph kernels from propagated information. Machine Learning, 102(2):209–245, 2016.

[54] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In Advances in neural information processing systems, pages 849–856, 2002.

[55] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In International Conference on Machine Learning, pages 2014–2023, 2016.

[56] Hoang NT and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters, 2019.

[57] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In International Conference on Learning Representations, 2020.

[58] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In International Conference on Learning Representations, 2020.

[59] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In International Conference on Learning Representations, 2020.

[60] Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs. IEEE transactions on signal processing, 61(7):1644–1656, 2013.

[61] Ryoma Sato. A survey on the expressive power of graph neural networks. arXiv preprint arXiv:2003.04078, 2020.

[62] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In Artificial Intelligence and Statistics, pages 488–495, 2009.

[63] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. IEEE signal processing magazine, 30(3):83–98, 2013.

[64] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3693–3702, 2017.

[65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008, 2017.

[66] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. International Conference on Learning Representations, 2018.

[67] Saurabh Verma and Zhi-Li Zhang. Hunt for the unique, stable, sparse and fast feature learning on graphs. In Advances in Neural Information Processing Systems, pages 88–98, 2017.

[68] Clément Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with structural message-passing. In Advances in Neural Information Processing Systems (NeurIPS) 33, pages 14143–14155. Curran Associates, Inc., 2020.

[69] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. Journal of Machine Learning Research, 11(Apr):1201–1242, 2010.
[70] 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.

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

[72] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[73] Zhang Xinyi and Lihui Chen. Capsule graph neural network. In *International Conference on Learning Representations*, 2019.

[74] Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. Graph wavelet neural network. In *International Conference on Learning Representations*, 2019.

[75] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[76] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462. PMLR, 2018.

[77] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374. ACM, 2015.

[78] Mingqi Yang, Yanming Shen, Heng Qi, and Baocai Yin. Breaking the expressible bottlenecks of graph neural networks. *arXiv preprint arXiv:2012.07219*, 2020.

[79] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation? *arXiv preprint arXiv:2106.05234*, 2021.

[80] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pages 4800–4810, 2018.

[81] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[82] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*, 2020.

[83] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[84] Shanzhuo Zhang, Lihang Liu, Sheng Gao, Donglong He, Xiaomin Fang, Weibin Li, Zhengjie Huang, Weiyue Su, and Wenjin Wang. Litegem: Lite geometry enhanced molecular representation learning for quantum property prediction, 2021.

[85] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: Algorithms, applications and open challenges. In *International Conference on Computational Social Networks*, pages 79–91. Springer, 2018.

[86] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gns. In *International Conference on Learning Representations*, 2020.

[87] Kaixiong Zhou, Xiao Huang, Yuening Li, Daochen Zha, Rui Chen, and Xia Hu. Towards deeper graph neural networks with differentiable group normalization. *arXiv preprint arXiv:2006.06972*, 2020.
[88] Hao Zhu and Piotr Koniusz. Simple spectral graph convolution. In *International Conference on Learning Representations*, 2020.

[89] Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. Interpreting and unifying graph neural networks with an optimization framework. In *Proceedings of the Web Conference 2021*, pages 1215–1226, 2021.
We use \( P = (p_1, p_2, \ldots, p_n) \). \( S = P \Lambda P^T \) is equivalent that \( S p_i = \lambda_i p_i, i \in [n] \). For any \( i \in [n] \), the geometric multiplicity of any \( \lambda_i \) is equal to its algebraic multiplicity, and \( E_{(S, \lambda_i)} = \text{Span}(\{p_k|\lambda_k = \lambda_i, k \in [n]\}) \). \( \hat{S} = P \phi(\Lambda) P^T \) and \( \hat{S} p_i = \phi(\lambda_i) p_i, i \in [n] \). Similarly, for any \( i \in [n] \), \( E_{(\hat{S}, \phi(\lambda_i))} = \text{Span}(\{p_k|\phi(\lambda_k) = \phi(\lambda_i), k \in [n]\}) \). Note that \( \{p_k|\lambda_k = \lambda_i, k \in [n]\} \subseteq \{p_k|\phi(\lambda_k) = \phi(\lambda_i), k \in [n]\} \) for any \( i \in [n] \). Hence \( \text{Span}(\{p_k|\lambda_k = \lambda_i, k \in [n]\}) \subseteq \text{Span}(\{p_k|\phi(\lambda_k) = \phi(\lambda_i), k \in [n]\}) \) As a result, \( E_{(S, \lambda_i)} \subseteq E_{(\hat{S}, \phi(\lambda_i))} \) for any \( i \in [n] \).

If \( \phi(\cdot) \) is injective, \( \{p_k|\lambda_k = \lambda_i, k \in [n]\} = \{p_k|\phi(\lambda_k) = \phi(\lambda_i), k \in [n]\} \) for any \( i \in [n] \). Thus \( E_{(S, \lambda_i)} = E_{(\hat{S}, \phi(\lambda_i))}. \)

We use \( \sigma(S) \) to denote the generalisation of the set of all eigenvalues of \( S \) (also known as the spectrum of \( S \)). Let \( S = P \Lambda_1 P^T \), \( B = Q \Lambda_2 Q^T \). If \( S \neq B \), then \( \hat{S} = F_\phi(S) \neq \hat{B} = F_\phi(B) \), we discuss two cases respectively.

**Case 1:** \( \sigma(S) \neq \sigma(B) \)

Then \( \sigma(\hat{S}) \neq \sigma(\hat{B}) \). The characteristic polynomials of \( \hat{S} \) and \( \hat{B} \) are different. Therefore, \( \hat{S} \neq \hat{B} \).

**Case 2:** \( \sigma(S) = \sigma(B) \)

Then \( \Lambda_1 = \Lambda_2 = \Lambda \). We prove the equivalent proposition \( \text{"} \hat{S} \neq \hat{B} \Rightarrow S = B \text{"} \). If \( \hat{S} = \hat{B} \), \( P \phi(\Lambda) P^T = Q \phi(\Lambda) Q^T \). For any \( \lambda_i \) with geometric multiplicity \( k \), we can find the corresponding eigenvectors \( p_1, p_2, \ldots, p_k \) according to \( P \phi(\Lambda) P^T \). Similarly, we can find the corresponding eigenvectors \( q_1, q_2, \ldots, q_k \) according to \( Q \phi(\Lambda) Q^T \). Note that the eigen-decomposition is unique in terms of eigenspaces. Thus, \( E_{(\hat{S}, \phi(\lambda_i))} = \text{Span}(p_1, p_2, \ldots, p_k) = \text{Span}(q_1, q_2, \ldots, q_k) = E_{(\hat{B}, \phi(\lambda_i))} \). Therefore, for any \( \lambda_i \), \( E_{(S, \lambda_i)} = E_{(B, \lambda_i)} \) (As given in Proposition 1.1). Correspondingly, \( S = P \Lambda P^T = Q \Lambda Q^T = B \). 

}\]
B Derivation of Eq. 9 and Eq. 10

\[
\cos(\langle Sh, p_i \rangle) = \frac{(Sh)^T p_i}{\|Sh\| |p_i|} = \frac{(Sh)^T p_i}{\|Sh\|} = \frac{(Sh)^T p_i}{\sqrt{(Sh)^T Sh}}
\]

\[
= \frac{(P \Lambda (P^T h))^T p_i}{\sqrt{(P \Lambda (P^T h))^T (P \Lambda (P^T h))}} = \frac{(P^T h)^T \Lambda P^T p_i}{\sqrt{(P^T h)^T \Lambda^2 (P^T h)}}
\]

\[
\begin{pmatrix} p_1^T \hfill \ldots \hfill p_i^T \hfill \ldots \hfill p_n^T \end{pmatrix} \begin{pmatrix} \lambda_1 \\
\vdots \\
\lambda_i \\
\vdots \\
\lambda_n \\
\end{pmatrix} \begin{pmatrix} p_1^T \\
\vdots \\
p_i^T \\
\vdots \\
p_n^T \\
\end{pmatrix} = \frac{p_i^T h \lambda_i}{\sqrt{\sum_{j=1}^{n} (x_j^T h)^2 \lambda_j^2}} = \frac{h^T p_i \lambda_i}{\sqrt{\sum_{j=1}^{n} (h^T x_j)^2 \lambda_j^2}}
\]

\[
\cos(\langle Sh, Sh' \rangle) = \frac{(Sh)^T Sh'}{\|Sh\| |Sh'|} = \frac{(Sh)^T Sh'}{\sqrt{(Sh)^T Sh} \sqrt{(Sh')^T Sh'}} = \frac{(P \Lambda (P^T h))^T P \Lambda (P^T h')}{\sqrt{(P \Lambda (P^T h))^T (P \Lambda (P^T h'))} \sqrt{(P \Lambda (P^T h'))^T (P \Lambda (P^T h'))}}
\]

\[
= \frac{(P^T h)^T \Lambda^2 P^T h'}{\sqrt{(P^T h)^T \Lambda^2 (P^T h')} \sqrt{(P^T h')^T \Lambda^2 (P^T h')}} = \frac{\alpha^T \Lambda^2 \beta}{\sqrt{\alpha^T \Lambda^2} \sqrt{\beta^T \Lambda^2} \beta} = \frac{\sum_{i=1}^{n} \alpha_i \beta_i \lambda_i^2}{\sqrt{\sum_{i=1}^{n} \alpha_i \lambda_i^2} \sqrt{\sum_{i=1}^{n} \beta_i \lambda_i^2}}
\]
C Proof of Proposition 2

Proof. (1) As $S^k = P \Lambda^k P^T$ and Eq. 8 we have

$$|\cos(\langle S^k h, p_1 \rangle)| = \frac{|\alpha_1 \lambda_1^k|}{\sqrt{\sum_{i=1}^n \alpha_i^2 \lambda_i^{2k}}}$$

Similarly, we can prove that $k$ must be convergent.

Note that when $k$ equals to 0,

$$\cos(\langle h, p_1 \rangle) = \frac{\alpha_1}{\sqrt{h^T h}} = \cos(\langle S^0 h, p_1 \rangle)$$

$$= \frac{\alpha_1}{\sqrt{\sum_{i=1}^n \alpha_i^2}}$$

Similarly, we can prove that $|\cos(\langle S^k h, p_n \rangle)| \geq |\cos(\langle S^{k+1} h, p_n \rangle)|$.

(2) Since $|\cos(\langle S^k h, p_n \rangle)|$ monotonously increases with respect to $k$ and has the upper bound 1, $|\cos(\langle S^k h, p_n \rangle)|$ must be convergent.

$$\lim_{k \to \infty} |\cos(\langle S^k h, p_1 \rangle)| = \lim_{k \to \infty} \frac{|\alpha_1 \lambda_1^k|}{\sqrt{\sum_{i=1}^n \alpha_i^2 \lambda_i^{2k}}}$$

$$= \lim_{k \to \infty} \frac{|\alpha_1|}{\sqrt{\alpha_1^2 + \sum_{i=2}^n \alpha_i^2 (\frac{\lambda_i}{\lambda_1})^{2k}}}$$

$$= \frac{|\alpha_1|}{\sqrt{\alpha_1^2 + \lim_{k \to \infty} \sum_{i=2}^n \alpha_i^2 (\frac{\lambda_i}{\lambda_1})^{2k}}}$$

As $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$, we have $\lim_{k \to \infty} \sum_{i=2}^n \alpha_i^2 (\frac{\lambda_i}{\lambda_1})^{2k} = 0$ and the convergent speed is dicided by $\frac{\alpha_2}{\lambda_1}$. Therefore $\lim_{k \to \infty} |\cos(\langle S^k h, p_1 \rangle)| = 1$. Similarly,

$$\lim_{k \to \infty} |\cos(\langle S^k h, S^k h' \rangle)| = \lim_{k \to \infty} \frac{|\sum_{i=1}^n \alpha_i \beta_i \lambda_i^{2k}|}{\sqrt{\sum_{i=1}^n \alpha_i^2 \lambda_i^{2k}} \sqrt{\sum_{i=1}^n \beta_i^2 \lambda_i^{2k}}}$$

$$= \lim_{k \to \infty} \frac{|\sum_{i=1}^n \alpha_i \beta_i \lambda_i^{2k}|}{\sqrt{\sum_{i=1}^n \alpha_i^2 \lambda_i^{2k}} \sqrt{\sum_{i=1}^n \beta_i^2 \lambda_i^{2k}}}$$

$$= \lim_{k \to \infty} \frac{|\alpha_1 \beta_1|}{\sqrt{\alpha_1^2 + \sum_{i=2}^n \alpha_i^2 (\frac{\lambda_i}{\lambda_1})^{2k}} \sqrt{\beta_1^2 + \sum_{i=2}^n \beta_i^2 (\frac{\lambda_i}{\lambda_1})^{2k}}}$$

$$= \frac{|\alpha_1 \beta_1|}{\sqrt{\alpha_1^2 \beta_1^2}}$$

$$= 1$$
D Architecture Overview

The overall architecture of non-spatial graph convolution is

\[ H^{(0)} = \text{MLP}_1(X_{\text{node attr}}) \in \mathbb{R}^{n \times d} \]
\[ E = \text{MLP}_2(X_{\text{edge attr}}) \in \mathbb{R}^{n \times n \times d} \]
\[ S = \text{MLP}_3(\sum_{i=0}^{k} S^{(i)}) \in \mathbb{R}^{n \times n \times d} \]

\[ H^{(k)} = H^{(k)} \cdot \text{copy}(0) \in \mathbb{R}^{n \times n \times d} \]
\[ H^{(k)} = \text{MLP}_4^{(k)}(H^{(k)} + E) \in \mathbb{R}^{n \times n \times d} \]
\[ H^{(k)} = H^{(k)} \odot S \in \mathbb{R}^{n \times n \times d} \]
\[ H^{(k)} = H^{(k)} \cdot \text{sum}(1) \in \mathbb{R}^{n \times d} \]
\[ H^{(k+1)} = \text{MLP}_5^{(k)}(H^{(k)}) \in \mathbb{R}^{n \times d} \]

E More Results
Figure 4: The curves of 5 runs with the number of basis $k = 9, 18, 21, 24$. 