CKGConv: General Graph Convolution with Continuous Kernels

Liheng Ma [1 2 3 †] **Soumyasundar Pal** [4] **Yitian Zhang** [1 2 3 †] **Jiaming Zhou** [4] **Yingxue Zhang** [4] **Mark Coates** [1 2 3]

Abstract

The existing definitions of graph convolution, either from spatial or spectral perspectives, are inflexible and not unified. Defining a general convolution operator in the graph domain is challenging due to the lack of canonical coordinates, the presence of irregular structures, and the properties of graph symmetries. In this work, we propose a novel and general graph convolution framework by parameterizing the kernels as continuous functions of pseudo-coordinates derived via graph positional encoding. We name this Continuous Kernel Graph Convolution (CKGConv). Theoretically, we demonstrate that CKGConv is flexible and expressive. CKGConv encompasses many existing graph convolutions, and exhibits a stronger expressiveness, as powerful as graph transformers in terms of distinguishing non-isomorphic graphs. Empirically, we show that CKGConv-based Networks outperform existing graph convolutional networks and perform comparably to the best graph transformers across a variety of graph datasets. The code and models are publicly available at `https://github.com/networkslab/CKGConv`.

1. Introduction

Recent advances in applying Transformer architectures in computer vision ignited a competition with the predominant Convolutional Neural Networks (ConvNets) (He et al., 2016a; Tan & Le, 2019). This rivalry started when Vision Transformers (ViTs) (Dosovitskiy et al., 2021; Wang et al., 2021; Liu et al., 2021; 2022a) exhibited impressive empirical gains over the best ConvNet architectures of the time.

[1]Department of ECE, McGill University, Montreal, Canada [2]Mila - Quebec AI Institute, Montreal, Canada [3]ILLS - International Laboratory on Learning Systems, Montreal, Canada [4]Huawei Noah's Ark Lab, Montreal, Canada [†]Work partially done as interns at Noah's Ark Lab. Correspondence to: Liheng Ma <liheng.ma@mail.mcgill.ca>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

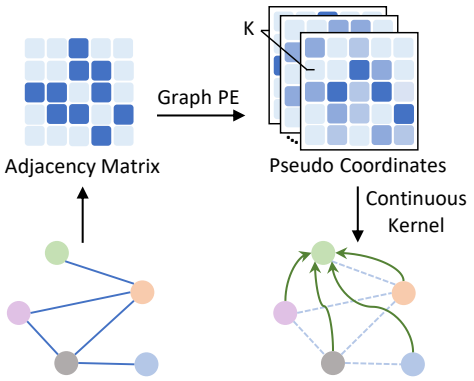

Figure 1. Continuous Kernel Graph Convolution (CKGConv)

However, several recent ConvNet variants (Liu et al., 2022b; Woo et al., 2023) achieve performance comparable to that of ViTs by incorporating innovative designs such as larger kernels and depthwise convolutions (Chollet, 2017).

In contrast, the appeal of Convolutional Graph Neural Networks (GNNs) seems to be diminishing; Graph Transformers (GTs) demonstrate elevated efficacy on many challenging graph learning tasks (Ying et al., 2021; Rampášek et al., 2022; Zhang et al., 2023; Ma et al., 2023). One reason might be that, unlike convolutions in Euclidean space, most existing definitions of graph convolution are inflexible and not unified. Message-passing Neural Networks (MPNNs) (Gilmer et al., 2017; Veličković, 2022) are defined in the spatial domain and limited to a one-hop neighborhood; Spectral GNNs (Bruna et al., 2014) are defined from a graph-frequency perspective and require careful designs (e.g., polynomial approximation (Defferrard et al., 2016; Wang & Zhang, 2022) or a sophisticated transformer-encoder (Bo et al., 2023)) to generalize to unseen graphs.

Unlike the general Euclidean convolution operators, there is no convolution operator for graphs that permits flexible determination of the support. Defining a general convolution operator in the graph domain is challenging due to the characteristics of a graph: the lack of canonical coordinates, the presence of irregular structures, and graph symmetries. These aspects are fundamentally different from Euclidean spaces. By addressing the aforementioned challenges, we generalize the continuous kernel convolution (Romero et al., 2022) to graph domain, and propose a general convolution

framework, namely *Continuous Kernel Graph Convolution* (CKGConv, as shown in Fig. 1). This subsumes several common convolutional GNNs, including non-dynamic[1] MPNNs, polynomial spectral GNNs, and diffusion-enhanced GNNs. We propose three designs to address the challenges of the graph domain: (1) pseudo-coordinates for graphs via positional encoding (addressing the lack of canonical coordinates and handling symmetries); (2) numerically stable scaled-convolution (compensating for irregular structure); (3) adaptive degree-scalers (mitigating the impact of symmetries).

CKGConv is endowed with several desirable properties. It exhibits immunity against over-smoothing and over-squashing. It encompasses Equivariant Set Neural Networks (Segol & Lipman, 2020) as a special case for the setting where there is no observed graph. Furthermore, we theoretically prove that CKGConv with generalized distance (GD) can be as powerful as Generalized Distance Weisfeiler-Lehman (GD-WL) (Zhang et al., 2023) in graph isomorphism tests (and is thus more powerful than 1-WL). Our experiments demonstrate the effectiveness of the proposed framework. CKGConv ranks among the best-performing models in a variety of graph learning benchmarks and significantly outperforms existing convolutional GNNs.

Our contributions are summarized as follows:

- We propose a novel graph convolution framework based on *pseudo-coordinates* of graphs and *continuous convolution kernels*.
- We demonstrate theoretically and empirically that our proposed graph convolution is expressively powerful, *outperforming existing convolutional GNNs* and achieving *comparable performance to the state-of-the-art graph transformers*.
- Various exploratory studies illustrate that CKGConv displays different and potentially complementary behavior *from graph transformers* that employ attention mechanisms. This motivates the combination of CKGConv and attention mechanisms as a potential direction towards designing more powerful graph models.

2. Related Work

Message-Passing Neural Networks *MPNNs* (Gilmer et al., 2017; Veličković, 2022) are a class of GNNs widely used in graph learning tasks. To update the representation of a node i, MPNNs aggregate the features from the direct neighbors of i. In most cases, the message-passing mechanisms can be viewed as convolution with kernels locally supported on a one-hop neighborhood. The kernels may be fixed (Kipf & Welling, 2017; Xu et al., 2019; Hamilton et al., 2017), learnable (Monti et al., 2017), or dynamic (Veličković et al., 2018; Bresson & Laurent, 2018). Recent *Graph Rewiring* techniques extend MPNNs beyond one-hop by introducing additional edges, guided by curvature (Topping et al., 2022), spectral gap (Arnaiz-Rodríguez et al., 2022), geodesic distance (Gutteridge et al., 2023), or positional encoding (PE) (Gabrielsson et al., 2023). Despite some similarity to our work, the usage of PE by Gabrielsson et al. (2023) is constrained by the MPNN framework and lacks flexibility.

Spectral and Polynomial Graph Neural Networks In contrast to MPNNs, *Spectral Graph Neural Networks* define graph filters in the spectral domain. The pioneering approach by Bruna et al. (2014) cannot generalize to an unseen graph with a different number of nodes. Follow-up works, constituting the class of *Polynomial Spectral Graph Neural Networks*, address this by (approximately) parameterizing the spectral filters by a polynomial of the (normalized) graph Laplacian (Defferrard et al., 2016; He et al., 2021; Liao et al., 2022; Wang & Zhang, 2022). Similarly, *Diffusion Enhanced Graph Neural Networks* extend spatial filters beyond one-hop by a polynomial of diffusion operators (e.g., adjacency matrix, random walk matrix) (Gasteiger et al., 2019b;a; Chien et al., 2021; Zhao et al., 2021; Frasca et al., 2020; Chamberlain et al., 2021), exhibiting strong connections to spectral GNNs. Notably, besides the polynomial approach, recent works endeavor to generalize spectral GNNs by introducing extra graph-order invariant operations on eigenfunctions of graph Laplacian (Beani et al., 2021; Bo et al., 2023).

Graph Transformers *GTs* are equipped with the transformer architecture (Vaswani et al., 2017), consisting of self-attention mechanisms (SAs) and feed-forward networks. Directly migrating the transformer architecture to graph learning tasks cannot properly utilize the topological information in graphs and leads to poor performance (Dwivedi & Bresson, 2021). Modern graph transformer architectures address this by integrating message-passing mechanisms (Kreuzer et al., 2021; Chen et al., 2022; Rampášek et al., 2022) or incorporating graph positional encoding (PE) (Kreuzer et al., 2021; Ying et al., 2021; Zhang et al., 2023; Ma et al., 2023). Appendix B provides more detail about graph PE.

Expressiveness of Graph Neural Networks Graph isomorphism tests have been widely used to measure the theoretical expressiveness of GNNs in terms of their ability to encode topological patterns. Without additional elements, MPNNs' expressive power is bounded by first-order Weisfeiler-Lehman (1-WL) algorithms (Xu et al., 2019). Polynomial spectral GNNs are as powerful as 1-WL algorithms; it is not known if this is a bound (Wang & Zhang, 2022). Higher-order GNNs (Morris et al., 2019) can reach

[1]We use the term *dynamic* to denote filters/kernels, generated dynamically conditioned on an input (Jia et al., 2016). *Filter* and *kernel* are used interchangeably in this work.

the same expressive power as K-WL with a cost of $O(N^K)$ computational complexity. Recently, Zhang et al. (2023) demonstrate that, under the $O(N^2)$ complexity constraint, Graph Transformers with generalized distance (GD) can go beyond 1-WL but are still bounded by 3-WL.

Continuous Kernel Convolution in Euclidean Spaces In order to handle irregularly sampled data and data at different resolutions, Romero et al. (2022) and Knigge et al. (2023) propose learning a convolution kernel as a continuous function, parameterized by a simple neural network, of the coordinates (relative positions), resulting in *Continuous Kernel Convolution*. This enables the convolution to generalize to any arbitrary support size with the same number of learnable parameters. Driven by different motivations, several works have explored similar ideas for *point cloud data* (Hermosilla et al., 2018; Wu et al., 2019; Xu et al., 2018; Hua et al., 2018). From a broader perspective, continuous kernels can be viewed as a subdomain of *Implicit Neural Representation* (Mildenhall et al., 2020; Sitzmann et al., 2020; Tancik et al., 2020), where the representation targets are the convolution kernels. Note that these techniques *rely on canonical coordinates* in Euclidean spaces and *cannot be directly applied to non-Euclidean domains* like graphs.

3. Methodology

3.1. Preliminary: Continuous Convolution Kernels

Let $x : \mathbb{Z} \to \mathbb{R}$ and $\psi : \mathbb{Z} \to \mathbb{R}$ be two scalar-valued real sequences sampled on the set of integers $\mathbb{Z}$, where $x[k]$ and $\psi[k]$ denote the signal and filter impulse response (kernel) at time k, respectively. The discrete convolution between the signal and the kernel at time k is defined as follows:

$$(x \star \psi)[k] := \sum_{\ell \in \mathbb{Z}} x[\ell]\psi[k-\ell]\,, \tag{1}$$

In most cases, the kernel is of finite width n_ψ, i.e., $\psi[k] = 0$ if $k < 0$ or $k \geqslant n_\psi$. The convolution sum in Eq. (1) is accordingly truncated at $\ell \notin [k - n_\psi + 1, k]$. However, learning such fixed support, discrete kernels cannot be generalized to arbitrary widths (i.e., different n_ψ) with the same set of parameters.

To address this shortcoming, Romero et al. (2022) propose to learn convolutional kernels by parameterizing $\psi[k]$ via a continuous function of k, implemented using a small neural network (e.g., multi-layer perception (MLP)). This is termed Continuous Kernel Convolution (CKConv). This formulation allows CKConv to model long-range dependencies and handle irregularly sampled data in Euclidean spaces.

3.2. Graph Convolution with Continuous Kernels

In the graph domain, convolution operators are required to handle varying sizes of supports, due to a varying number of nodes and the irregular structures. We explore the potential of continuous kernels for graphs and propose a general graph convolution framework with continuous kernels.

The generalization of continuous kernels to graph domain is not trivial due to the following characteristics: (1) the *lack of canonical coordinates* (Bruna et al., 2014) makes it difficult to define the relative positions between non-adjacent nodes; (2) the *irregular structure* requires the kernel to generalize to different support sizes while retaining numerical stability (Veličković et al., 2020; Corso et al., 2020); (3) the presence of *graph symmetries* demands that the kernel can distinguish between nodes in the support without introducing permutation-sensitive operations (Hamilton et al., 2017) or extra ambiguities (Lim et al., 2023).

These challenges drive us to propose a general graph convolution framework with continuous kernels, namely CKGConv. Our overall design consists of three innovations: (1) we use graph positional encoding to derive the pseudo-coordinates of nodes and define relative positions; (2) we introduce a scaled convolution to handle the irregular structure; (3) we incorporate an adaptive degree-scaler to improve the representation of structural graph information.

3.2.1. Graph Positional Encoding as Pseudo-coordinates

In contrast to Euclidean spaces, the graph domain is known to lack canonical coordinates. Consequently, it is not trivial to define the relative distance between two non-adjacent nodes. Pioneering work (Monti et al., 2017) attempted to define *pseudo-coordinates* for graphs, however, the definition was restricted to one-hop neighborhoods.

In this work, we reveal that pseudo-coordinates can be naturally defined by graph positional encoding (PE), allowing us to specify relative positions for continuous kernels beyond the one-hop neighborhood constraint. Specifically, we use Relative Random Walk Probabilities (RRWPs) (Ma et al., 2023), which have been demonstrated to be one of the most expressive graph positional encodings (Black et al., 2024). Let $\mathbf{A} \in \mathbb{R}^{N \times N}$ be the adjacency matrix of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes, and let $\mathbf{D}$ be the diagonal degree matrix, $\mathbf{D}_{i,i} = \sum_{j \in \mathcal{V}} \mathbf{A}_{i,j}$. The random walk matrix is $\mathbf{M} := \mathbf{D}^{-1}\mathbf{A}$. Entry $\mathbf{M}_{ij}$ is then the probability of a move from node i to node j in one step of a simple random walk. The (top) K-RRWP for each pair of nodes $i, j \in \mathcal{V}$ consists of zeroth to $(K-1)$th powers of random walk matrix, defined as:

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

where $\mathbf{I} \in \mathbb{R}^{N \times N}$ denoting the identity matrix. We add an extra re-scaling on RRWP to remove the dependency on graph-orders (details in Appendix A.2).

RRWP is not the only choice for constructing pseudo-coordinates. One can use other graph positional encodings such as shortest-path-distance (SPD) (Ying et al., 2021) and resistance distance (RD) (Zhang et al., 2023).

3.2.2. Numerically Stable Graph Convolution

When applying a kernel to different nodes in graphs, the support size can vary remarkably. To ensure numerical stability and the ability to generalize, it is crucial to avoid disproportionate scaling of different node representations (Veličković et al., 2020).

Therefore, we introduce a scaling term to perform scaled convolution in CKGConv. We consider a kernel function $\psi : \mathbb{R}^r \to \mathbb{R}$. For a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node-signal function $\chi : \mathcal{V} \to \mathbb{R}$, CKGConv is defined as:[2]

$$(\chi \star \psi)(i) := \frac{1}{|\operatorname{supp}_\psi(i)|} \sum_{j \in \operatorname{supp}_\psi(i)} \chi(j) \cdot \psi(\mathbf{P}_{i,j}) + b \,. \quad (3)$$

Here $b \in \mathbb{R}$ is a learnable bias term; the set $\operatorname{supp}_\psi(i)$ is the predefined support of kernel ψ for node i (i.e., $|\operatorname{supp}_\psi(i)|$ denotes the kernel size); and $\mathbf{P}_{i,j} \in \mathbb{R}^K$ is the relative positional encoding.

Owing to the flexibility of CKGConv, we can set $\operatorname{supp}_\psi(i)$ to be the K-hop neighborhood[3] of node i, with K being an arbitrary positive integer. Alternatively, we can choose the support to be the entire graph, thereby constructing a global kernel. This flexibility arises because the construction of pseudo-coordinates is decoupled from the evaluation of the convolution kernel. We show that the globally supported variant is endowed with several desired theoretical properties (Sec. 3.3 and Sec. 4.2).

3.2.3. Depthwise Separable Convolution

We extend the scalar-valued definition of CKGConv (shown in Eq. (3)) to vector-valued signals ($\boldsymbol{\chi} : \mathcal{V} \to \mathbb{R}^d$ and $(\boldsymbol{\chi} \star \boldsymbol{\psi})(i) : \mathcal{V} \to \mathbb{R}^{d'}$) via the Depthwise Separable Convolution (DW-Conv) architecture (Chollet, 2017),

$$(\boldsymbol{\chi} \star \boldsymbol{\psi})(i) := \mathbf{W}\Big(\tfrac{1}{|\operatorname{supp}_\psi(i)|} \sum_{j \in \operatorname{supp}_\psi(i)} \boldsymbol{\chi}(j) \odot \boldsymbol{\psi}(\mathbf{P}_{i,j})\Big) + \mathbf{b} \,. \quad (4)$$

Here $\boldsymbol{\psi} : \mathbb{R}^K \to \mathbb{R}^d$ is a kernel function acting on a vector; $\mathbf{W} \in \mathbb{R}^{d' \times d}$ and $\mathbf{b} \in \mathbb{R}^{d'}$ are the learnable weights and bias, respectively, shared by all nodes; and $\odot$ stands for elementwise multiplication.

We can alternatively extend Eq. (3) to multiple channels via grouped convolution (Krizhevsky et al., 2012), multi-head architectures (Vaswani et al., 2017), or even MLP-Mixer architectures (Tolstikhin et al., 2021; Touvron et al., 2023). We select DWConv because it provides a favorable trade-off between expressiveness and the number of parameters.

3.2.4. MLP-based Kernel Function

In this work, as an example, we introduce kernel functions parameterized by multi-layer perceptrons (MLPs), but the proposed convolution methodology accommodates many other kernel functions. Each MLP block consists of fully connected layers (`FC`), non-linear activation (σ), a normalization layer (`norm`) and a residual connection, inspired by ResNetv2 (He et al., 2016b):

$$\texttt{MLP}(\mathbf{x}) := \mathbf{x} + \texttt{FC} \circ \sigma \circ \texttt{Norm} \circ \texttt{FC} \circ \sigma \circ \texttt{Norm}(\mathbf{x}) \,. \quad (5)$$

Here $\circ$ denotes function composition; $\texttt{FC}(\mathbf{x}) := \mathbf{W}\mathbf{x} + \mathbf{b}$, with learnable weight matrix, $\mathbf{W} \in \mathbb{R}^{r \times r}$, and bias, $\mathbf{b} \in \mathbb{R}^r$; and we use GELU (Hendrycks & Gimpel, 2023) as the default choice of σ.

The overall kernel function $\boldsymbol{\psi}$ is defined as:

$$\boldsymbol{\psi}(\mathbf{P}_{\mathbf{i},\mathbf{j}}) := \texttt{FC} \circ \texttt{Norm} \circ \texttt{MLP} \circ \cdots \circ \texttt{MLP}(\mathbf{P}_{i,j}) \,, \quad (6)$$

where the last $\texttt{FC} : \mathbb{R}^r \to \mathbb{R}^d$ maps to the desired number of output channels.

3.2.5. Degree Scaler

As a known issue in graph learning, the scaled convolutions and mean-aggregations cannot properly preserve the degree information of nodes (Xu et al., 2019; Corso et al., 2020). Therefore, we introduce a post-convolution adaptive degree-scaler into the node representation to recover such information, following the approach proposed by Ma et al. (2023):

$$\mathbf{x}'_i := \mathbf{x}_i \odot \boldsymbol{\theta}_1 + \left(d_i^{1/2} \cdot \mathbf{x}_i \odot \boldsymbol{\theta}_2\right) \in \mathbb{R}^d \,. \quad (7)$$

Here $d_i \in \mathbb{R}$ is the degree of node i, and $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^r$ are learnable weight vectors.

As an alternative, we also introduce a variant that injects the degree information directly into the RRWP, $\mathbf{P}_{i,j} \in \mathbb{R}^r$,

[2] In the Euclidean domain, conventional *convolution* involves *reversal* and *shifting* of the filter (kernel). The meaning of reversal is not obvious in the graph domain. Although there is no explicit reversal in our procedure, the kernel ψ is a mapping from a positive relative positional encoding $\mathbf{P}_{i,j}$. For each node i, we can thus view it as a symmetric filter with respect to a corresponding absolute positional encoding (that we do not specify), and reversal would not change the filtering coefficient for a node j. We therefore retain the usage of the terminology *convolution* in this work.

[3] The K-hop neighborhood of node i is the set of nodes whose shortest-path distance from node i is smaller than or equal to K.

before applying the kernel function ψ:

$$\hat{\mathbf{P}}_{i,j} := \mathbf{P}_{i,j} \odot \boldsymbol{\theta}_1 + \Big((d_i^{1/2} \odot \boldsymbol{\theta}_2) \odot \mathbf{P}_{i,j} \odot (d_j^{-1/2} \odot \boldsymbol{\theta}_3) \Big) \,, \tag{8}$$

where $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3 \in \mathbb{R}^r$ are learnable parameters and $\hat{\mathbf{P}}_{i,j}$ is used instead of $\mathbf{P}_{i,j}$ in other parts. This variant enjoys several desired theoretical properties as discussed in Sec. 4.3. However, in practice, we did not observe any significant differences in empirical performance, and use Eq. (7) in our experiments due to its computational efficiency.

3.2.6. Overall Architecture of CKGCN

The overall multi-layer architecture of the proposed model, denoted by Continuous Kernel Graph Convolution Network (CKGCN), consists of L CKGConv-blocks as the backbone, together with task-dependent output heads (as shown in Fig. 3 in Appendix A). Each CKGConv block consists of a CKGConv layer and a feed-forward network (FFN), with residual connections and a normalization layer:

$$\texttt{CKGConvBlock}(\cdot) := \texttt{norm} \circ \texttt{FFN} \circ \texttt{norm} \circ \texttt{CKGConv}(\cdot) \,. \tag{9}$$

We use BatchNorm (Ioffe & Szegedy, 2015) in the main branch as well as in the kernel functions. Using LayerNorm (Ba et al., 2016) has the potential to cancel out the degree information (Ma et al., 2023). Appendix E.4 presents additional architectural details. The input node/edge attributes ($\mathbf{x}'_i \in \mathbb{R}^{d_h}$ and $\mathbf{e}'_{i,j} \in \mathbb{R}^{d_e}$) and the absolute/relative positional encoding (RRWP) are concatenated: $\mathbf{x}_i = [\mathbf{x}'_i \| \mathbf{P}'_{i,i}] \in \mathbb{R}^{d_h+K}$ and $\mathbf{P}_{i,j} = [\mathbf{e}'_{i,j} \| \mathbf{P}'_{i,j}] \in \mathbb{R}^{d_e+K}$, where $\mathbf{P}'_{i,j}$ denotes the input PE. A linear projection (a *stem*) maps to the desired dimensions before the backbone. If the data does not include node/edge attributes, zero-padding is used. For a fair comparison, we use the same task-dependent output heads as previous work (Rampášek et al., 2022).

3.3. Theory: CKGCN Is as Expressive as Graph Transformers

Zhang et al. (2023) prove that Graph Transformers with generalized distance (GD) can be as powerful as GD-WL with a proper choice of attention mechanisms, thus going beyond 1-WL and bounded by 3-WL. We provide a similar constructive proof, demonstrating that CKGConv with GD is as powerful as GD-WL. It achieves the same theoretical expressiveness as SOTA graph transformers (Zhang et al., 2023; Ma et al., 2023), with respect to the GD-WL test.

Proposition 3.1. *A Continuous Kernel Graph Convolution Network (CKGCN), stacking feed-forward networks (FFNs) and globally supported CKGConvs with generalized distance (GD) as pseudo-coordinates, is as powerful as GD-WL, when choosing the proper kernel $\boldsymbol{\psi}$.*

The proof is provided in Appendix E.1.

4. Relationship with Previous Work

4.1. Beyond the Limitations of MPNNs

Despite being widely used, MPNNs are known to exhibit certain limitations: (1) over-smoothing; (2) over-squashing and under-reaching; (3) expressive power limited to 1-WL. By contrast, CKGConv inherently addresses these constraints.
Over-smoothing (Li et al., 2018; Oono & Suzuki, 2020) arises because most MPNNs apply a smoothing operator (a blurring kernels or low-pass filter). CKGConv can generate sharpening kernels, and thus does not suffer from over-smoothing, as illustrated in a toy example in Appendix D.1.
Over-squashing (Alon & Yahav, 2020; Topping et al., 2022) is mainly due to the local message-passing within the one-hop neighborhood. The kernels in CKGConv can have supports beyond one-hop neighborhoods. Both the empirical performance on the Long-Range Graph Benchmark (Dwivedi et al., 2022c) shown in Table 2 and the ablation study in Appendix D.3 showcase the effect of expanding the supports and indicate the necessity to go beyond local message-passing.
Regarding the ***expressiveness*** (Xu et al., 2019; Morris et al., 2019; Loukas, 2020), we have demonstrated in Sec. 3.3 that CKGConv can reach expressive power equivalent to GD-WL, thus going beyond 1-WL algorithms. The empirical experiments also validate the capacity of CKGConv.

4.2. Equivariant Set Neural Networks

In general, graphs can be viewed as a set of nodes with observed structures among them. Here, we demonstrate that when the pseudo-coordinates do not encode any graph structure, CKGConv can naturally degenerate to the general form of a layer in an *Equivariant Set Network* (Segol & Lipman, 2020). This matches the natural transition between graph data and set data. The following proposition states that when we use 1-RRWP (i.e., an Identity matrix) as the pseudo-coordinate (and thus ignore any graph structure), CKGConv is equivalent to a layer of an equivariant set network.

Proposition 4.1. *With 1-RRWP $\mathbf{P} = [\mathbf{I}]$, CKGConv with a globally supported kernel can degenerate to the following general form of a layer in an Equivariant Set Network (Eq. (8) in Segol & Lipman (2020)):*

$$(\chi \star \psi)(i) = \gamma \cdot \chi(i) + \beta \cdot \big(\frac{1}{|\mathcal{V}|} \sum_{j \in \mathcal{V}} \chi(j) \big) + b \,. \tag{10}$$

Here $\gamma, \beta, b \in \mathbb{R}$ are learnable parameters. This can be directly generalized to vector-valued signals.

The proof is provided in Appendix E.2.

4.3. Polynomial Spectral and Diffusion Enhanced GNNs

Polynomial spectral GNNs approximate spectral convolution by fixed-order polynomial functions of the symmetric normalized graph Laplacian matrix $\tilde{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \in \mathbb{R}^{n\times n}$. Similarly, diverse *Diffusion Enhanced GNNs* use polynomial parameterization with the diffusion operator $\tilde{\mathbf{A}}$ or $\mathbf{M}$ replacing the graph Laplacian.

The following proposition states that CKGConv can represent any polynomial spectral GNN or diffusion-enhanced GNN of any order with suitable injections of the node degrees.

Proposition 4.2. *With K-RRWP $\mathbf{P}_{i,j} \in \mathbb{R}^K$ as pseudo-coordinates, CKGConv with a linear kernel ψ can represent any* Polynomial Spectral GNN *or any* Diffusion Enhanced GNNs *of $(K-1)$th order exactly, regardless of the specific polynomial parameterization, if degree $d_i^{1/2}$ and $d_j^{-1/2}$ are injected to $\mathbf{P}_{i,j}$ properly.*

The proof is presented in Appendix E.3. If $K \to \infty$, CKGConv can closely approximate a full spectral GNN. This also highlights the relationship between RRWP and graph spectral wavelets (Hammond et al., 2011).

Note that CKGConv is strictly *more expressive* than previous polynomial spectral GNNs and diffusion-enhanced GNNs. While polynomial spectral GNNs and diffusion-enhanced GNNs are constrained by the linear combinations of powers of Laplacian/diffusion operators, CKGConv, equipped with non-linear kernels ψ such as MLPs, can construct more general convolution kernels. Since MLPs are universal function approximators (Hornik et al., 1989), CKGConv can represent a considerably richer class of functions. The ablation study on the kernel functions (in Appendix D.4) also verifies the importance of introducing kernel functions beyond linear transformations.

4.4. Fourier Features of Graphs

As mentioned in Proposition 4.2, RRWP can be viewed as a set of bases for a polynomial vector space, which approximates the full Fourier basis of graphs (i.e., eigenvectors of the Laplacian). Therefore, RRWP can be viewed as a set of (approximate) Fourier features under certain transformations. Likewise in the Euclidean space, Tancik et al. (2020) propose to construct *Fourier features* from coordinates to let MLPs better capture high-frequency information.

Another existing approach broadly related to our work is the Specformer (Bo et al., 2023), which generates graph spectral filters via transformers, given a sampled collection of Fourier bases[4] in the spectral domain. Specformer approximates the full Fourier bases from the spectral perspective, whereas CKGConv performs an approximation in the spatial domain. In a similar fashion to the contrast between the Fourier transform and the wavelet decomposition, Specformer achieves better localization on frequencies, and CKGConv exhibits better localization spatially.

[4]Operating on the full Fourier bases has $O(N^3)$ computational complexity, where N is the number of nodes in the graph.

4.5. Graph Transformers

As shown in Sec. 3.3, with the same generalized distances (e.g., SPD, RD, RRWP) as relative positional encoding or pseudo-coordinates, CKGConv can reach the same theoretical expressive power as Graph Transformers, with respect to graph isomorphism tests.

From a filtering perspective, self-attention in (Graph) Transformers can be viewed as a dynamic filter (Park & Kim, 2021). However, the filter coefficients are constrained to be positive, and thus self-attention can only perform blurring or low-pass filtering. In contrast, CKGConv is a non-dynamic filter, but has the flexibility to include positive and negative coefficients simultaneously and thus can generate sharpening kernels.

In this work, we do not claim that CKGConv is better than Graph Transformers, or vice versa. We emphasize that each approach has its own advantages. The contrasting strengths of dynamic and sharpening, present an intriguing possibility of developing architectures that combine the strengths of graph transformers and continuous convolution. Exploratory experiments in Sec. 5.4 highlight the behavioral differences between graph transformers and CKGCNs, and examine the performance of a preliminary, naive combination.

5. Experimental Results

5.1. Benchmarking CKGCN

We evaluate our proposed method on five datasets from *Benchmarking GNNs* (Dwivedi et al., 2022a) and another two datasets from *Long-Range Graph Benchmark* (Dwivedi et al., 2022c). These benchmarks include diverse node- and graph-level learning tasks such as node classification, graph classification, and graph regression. They test an algorithm's ability to focus on graph structure encoding, to perform node clustering, and to learn long-range dependencies. The statistics of these datasets and further details of the experimental setup are deferred to Appendix C.

Baselines We compare our methods with

- *SOTA Graph Transformer:* GRIT (Ma et al., 2023);
- *Hybrid Graph Transformer (MPNN+self-attention):* GraphGPS (Rampášek et al., 2022);
- *Popular Message-passing Neural Networks:* GCN (Kipf & Welling, 2017), GIN (Xu et al., 2019) and its variant with edge-features (Hu et al., 2020), GAT (Veličković et al., 2018), GatedGCN (Bresson &

Table 1. Test performance in five benchmarks from (Dwivedi et al., 2022a; Ma et al., 2023; Bo et al., 2023). Shown is the mean ± s.d. of 4 runs with different random seeds. Highlighted are the top **first**, **second**, and **third** results. # Param under $500K$ for ZINC, PATTERN, CLUSTER and $\sim 100K$ for MNIST and CIFAR10.

Model	ZINC	MNIST	CIFAR10	PATTERN	CLUSTER
	MAE↓	Accuracy↑	Accuracy↑	W. Accuracy↑	W. Accuracy↑
GCN	0.367 ± 0.011	90.705 ± 0.218	55.710 ± 0.381	71.892 ± 0.334	68.498 ± 0.976
GIN	0.526 ± 0.051	96.485 ± 0.252	55.255 ± 1.527	85.387 ± 0.136	64.716 ± 1.553
GAT	0.384 ± 0.007	95.535 ± 0.205	64.223 ± 0.455	78.271 ± 0.186	70.587 ± 0.447
GatedGCN	0.282 ± 0.015	97.340 ± 0.143	67.312 ± 0.311	85.568 ± 0.088	73.840 ± 0.326
GatedGCN-LSPE	0.090 ± 0.001	–	–	–	–
PNA	0.188 ± 0.004	97.94 ± 0.12	70.35 ± 0.63	–	–
GSN	0.101 ± 0.010	–	–	–	–
DGN	0.168 ± 0.003	–	72.838 ± 0.417	86.680 ± 0.034	–
Specformer	$\mathbf{0.066 \pm 0.003}$	-	-	-	-
CIN	0.079 ± 0.006	–	–	–	–
CRaW1	0.085 ± 0.004	97.944 ± 0.050	69.013 ± 0.259	–	–
GIN-AK+	0.080 ± 0.001	–	72.19 ± 0.13	$\mathbf{86.850 \pm 0.057}$	–
SAN	0.139 ± 0.006	–	–	86.581 ± 0.037	76.691 ± 0.65
Graphormer	0.122 ± 0.006	–	–	–	–
K-Subgraph SAT	0.094 ± 0.008	–	–	86.848 ± 0.037	77.856 ± 0.104
EGT	0.108 ± 0.009	$\mathbf{98.173 \pm 0.087}$	68.702 ± 0.409	86.821 ± 0.020	$\mathbf{79.232 \pm 0.348}$
Graphormer-URPE	0.086 ± 0.007	–	–	–	–
Graphormer-GD	0.081 ± 0.009	–	–	–	–
GPS	$\mathbf{0.070 \pm 0.004}$	98.051 ± 0.126	72.298 ± 0.356	86.685 ± 0.059	78.016 ± 0.180
GRIT	$\mathbf{0.059 \pm 0.002}$	$\mathbf{98.108 \pm 0.111}$	$\mathbf{76.468 \pm 0.881}$	$\mathbf{87.196 \pm 0.076}$	$\mathbf{80.026 \pm 0.277}$
CKGCN	$\mathbf{0.059 \pm 0.003}$	$\mathbf{98.423 \pm 0.155}$	$\mathbf{72.785 \pm 0.436}$	$\mathbf{88.661 \pm 0.143}$	$\mathbf{79.003 \pm 0.140}$

Laurent, 2018), GatedGCN-LSPE (Dwivedi et al., 2022b), and PNA (Corso et al., 2020);

- *Other Graph Transformers:* Graphormer (Ying et al., 2021), K-Subgraph SAT (Chen et al., 2022), EGT (Hussain et al., 2022), SAN (Kreuzer et al., 2021), Graphormer-URPE (Luo et al., 2022), and Graphormer-GD (Zhang et al., 2023);
- *SOTA Spectral Graph Neural Networks:* Specformer (Bo et al., 2023) and DGN (Beani et al., 2021); and
- *Other SOTA Graph Neural Networks:* GSN (Bouritsas et al., 2022), CIN (Bodnar et al., 2021), CRaW1 (Tönshoff et al., 2023), and GIN-AK+ (Zhao et al., 2022).

Benchmarks from Benchmarking GNNs In Table 1, we report the results on five datasets from (Dwivedi et al., 2022a): ZINC, MNIST, CIFAR10, PATTERN, and CLUSTER. We observe that the proposed CKGConv achieves the best performance for 3 out of 5 datasets and is ranked within the three top-performing models for the other 2 datasets. Compared to the hybrid transformer, *GraphGPS*, consisting of an MPNN and self-attention modules (SAs), CKGCN outperforms on all five datasets, indicating the advantage of the continuous convolution employed by CKGConv over MPNNs, even when enhanced by self-attention. *GRIT* and CKGConv achieve comparable performance but exhibit advantage in different datasets. CKGConv outperforms on MNIST and PATTERN while GRIT performs better on CIFAR10 and CLUSTER. This suggests that the capability to learn *dynamic* kernels and *sharpening* kernels might have different impact and value on the empirical performance depending on the nature of the dataset. Notably, CKGConv exhibits superior performance compared to all other convolution-based GNNs for four of the five datasets. The only exception is CIFAR10, where CKGConv is slightly worse than DGN, although the difference is not statistically significant. [5]

Long-Range Graph Benchmark (LRGB) Graph transformers demonstrate advantages over MPNNs in modeling long-range dependencies. Here, we verify the capacity of CKGConv to model long-range dependencies. We conduct experiments on two peptide graph datasets from the Long-Range Graph Benchmark (LRGB) (Dwivedi et al., 2022c). The obtained results are summarized in Table 2. On both datasets, CKGConv obtains the second-best mean performance. Based on a two-sample one-tailed t-test, the performance is not significantly different from the best-performing algorithm (GRIT). There is, however, a statistically significant difference between CKGConv's performance and the third-best algorithm's performance for both datasets. This demonstrates that our model is able to learn long-range

[5] According to a two-sided t-test at the 5% significance level.

Table 2. Test performance on two benchmarks from long-range graph benchmarks (LRGB) (Dwivedi et al., 2022c). Shown is the mean ± s.d. of 4 runs with different random seeds. Highlighted are the top **first**, **second**, and **third** results. # Param $\sim 500K$.

Model	Peptides-func	Peptides-struct
	AP↑	MAE↓
GCN	0.5930 ± 0.0023	0.3496 ± 0.0013
GINE	0.5498 ± 0.0079	0.3547 ± 0.0045
GatedGCN	0.5864 ± 0.0035	0.3420 ± 0.0013
GatedGCN+RWSE	0.6069 ± 0.0035	0.3357 ± 0.0006
Transformer+LapPE	0.6326 ± 0.0126	0.2529 ± 0.0016
SAN+LapPE	0.6384 ± 0.0121	0.2683 ± 0.0043
SAN+RWSE	0.6439 ± 0.0075	0.2545 ± 0.0012
GPS	$\mathbf{0.6535 \pm 0.0041}$	$\mathbf{0.2500 \pm 0.0012}$
GRIT	$\mathbf{0.6988 \pm 0.0082}$	$\mathbf{0.2460 \pm 0.0012}$
CKGCN	$\mathbf{0.6952 \pm 0.0068}$	$\mathbf{0.2477 \pm 0.0018}$

interactions, on par with the SOTA graph transformers.

5.2. The Flexible Kernels of CKGConv

Convolutional kernels with both negative and positive coefficients have a long history. Such kernels are widely used to amplify the signal differences among data points, e.g., signal sharpening and edge detection in image processing. Here, we highlight that CKGConv has the flexibility to generate kernels that include negative and positive coefficients.

5.2.1. CKGConv Kernel Visualization

We show that CKGConv can learn positive and negative kernel coefficients from the data, without being forced to generate negative kernel coefficients. Therefore, we visualize the learned kernels of CKGConv from real-world graph learning tasks. Specifically, we visualize two selected learned kernels from the depthwise convolution of CKGConv for each of the two graphs from the ZINC datasets, as shown in Fig. 2. Several learned kernels in CKGConv indeed generate both positive and negative coefficients.

5.2.2. Ablation Study on Flexible Kernels

To showcase the importance of the flexible kernels in CKGConv on graph learning tasks, we conduct an ablation study on ZINC and PATTERN, comparing CKGCN to its blurring-kernel variant, which is constrained to generate all-positive coefficients by incorporating a Softmax operation.

From Table 3, constraining CKGCN to generate blurring kernels leads to remarkable performance deterioration. In addition, GRIT with self-attention (SA) mechanisms, which can be viewed as *dynamic* blurring kernels (Park & Kim, 2021), outperforms CKGCN-Blurring. This observation

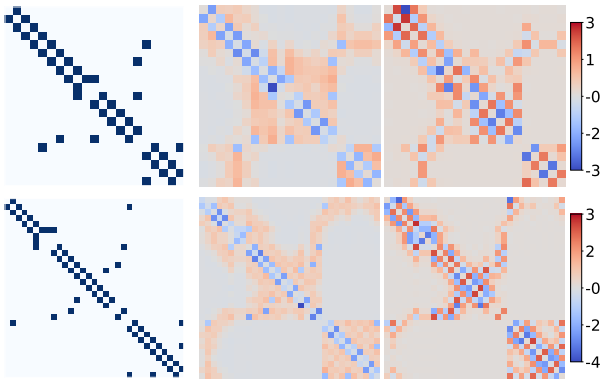

Figure 2. Adjacency matrices and learned continuous kernels across multiple channels for two graphs from the ZINC dataset.

Table 3. Effect of different kernel types on ZINC and PATTERN.

Model	ZINC	PATTERN	*Dynamic*	*Flexible*
	MAE↓	W. Acc.↑		
GRIT	0.059 ± 0.002	87.196 ± 0.076	✓	✗
CKGCN	0.059 ± 0.003	88.661 ± 0.143	✗	✓
-Blurring	0.073 ± 0.003	87.000 ± 0.002	✗	✗

indicates that both dynamic and flexible properties are beneficial to graph learning and hints at the potential combination of CKGConvs and SAs for graph learning.[6]

5.3. Sensitivity Study on the Choice of Graph PEs

In this paper, we demonstrate the efficacy of CKGConv with RRWP to avoid PE becoming the bottleneck of the model performance. However, CKGConv is not constrained to working with a specific graph PE. As depicted in Proposition E.1, the choices of PE affect the expressive power of CKGConv, depending on the structural/positional information encoded. Therefore, in this section, we study the impact of using different graph PEs, and demonstrate that CKConv can reach a competitive performance with other well-designed and expressive PEs besides RRWP.

We conduct the sensitivity study on ZINC datasets with four typical graph PEs: RRWP (Ma et al., 2023), Resistance Distance (RD) (Zhang et al., 2023), Shortest-path distance (SPD) (Ying et al., 2021), and "Pair-RWSE", which is constructed as relative PE by concatenating the Random Walk Structural Encoding (RWSE) (Dwivedi et al., 2022b) for each node-pair. We add RWSE as the absolute PE to the node attribute when using other PEs, mimicking RRWP. The experimental setup follows the main experiment and the results of 4 runs are reported in Table 4.

The results of SPD and Pair-RWSE show that a sub-optimal

[6]Note that attention mechanisms typically require the incorporation of Softmax to stabilize the attention scores.

Table 4. The sensitive study of CKGCN on the choices of graph PEs. Shown is the mean ± s.d. of 4 runs.

CKGCN	RRWP	RD	SPD	Pair-RWSE
MAE ↓	0.059 ± 0.003	0.062 ± 0.004	0.072 ± 0.003	0.081 ± 0.002

PE design leads to worse performance of CKGCN. However, with an expressive PE, CKGCN demonstrates stable performance: CKGCN with either RD or RRWP achieves competitive performance that is statistically indistinguishable from the state-of-the-art.[7]

5.4. CKGCN and GTs Behave Differently

Motivated by the observation in Sec. 5.2.2 and previous work on ViT (Park & Kim, 2021), in this section, we aim to demonstrate that CKGCNs and graph transformers learn complementary features in graph learning. Thus, we conduct an ensembling experiment on ZINC to examine the effects of naively combining a SOTA graph transformer GRIT with CKGCN.

In Table 5, we report the mean and standard deviation of MAE (employing bootstrapping) of an ensemble of GRIT models, an ensemble of CKGConv models, and a mixed ensemble using both of these models. In each case, the total ensemble size is 4 (two of each model for the mixed ensemble). We observe that constituting the ensemble using both CKGConv and SAs offers a statistically significant advantage compared to either homogeneous ensemble.

Table 5. Effect of ensembling on ZINC.

Model	**GRIT-Ens.**	**CKGConv-Ens.**	**Mixed-Ens.**
MAE ↓	0.054±0.001	0.054±0.002	**0.051±0.001***

Based on the observation, we hypothesize that both CKGConvs and SAs have their own merits, and it can be further advantageous to suitably combine them in the model architecture. Similar efforts have been undertaken in computer vision (Park & Kim, 2021; Xiao et al., 2021).

5.5. Further Analyses: Anti-Oversmoothing, Edge-detection, Support Sizes and Kernel Functions

We include the results of additional experiments to further analyse the performance and behavior of CKGConv in Appendix D. Specifically, we include:

- Two toy examples that demonstrate the advantages of (positive and) negative kernel coefficients.
  - Appendix D.1 showcases that CKGConv effectively counters *oversmoothing*.
  - Appendix D.2 demonstrates the efficacy of CKGConv for *edge-detection*.[8]
- Two ablation/sensitivity studies on the kernel designs.
  - Appendix D.3 studies the impact of the *support size*, which demonstrates the utility of localized kernels and highlights the importance of going *beyond local message-passing*.
  - Appendix D.4 analyzes the impact from the *number of MLP blocks* in the kernel functions, which indicates the necessity of *non-linear kernel functions*.

6. Limitations

On the computational side, a naive implementation of CKGConv with global support has $O(|\mathcal{V}|^2)$ complexity, the same as graph transformers. A more efficient alternative implementation is provided in Appendix C.7, which might prevent the usage of some operators such as BatchNorm. The localized CKGConv can benefit from lower computation complexity but with weaker theoretical expressiveness.

7. Conclusion

Motivated by the lack of a flexible and powerful convolution mechanism for graphs, we propose a general graph convolution framework, CKGConv, by generalizing continuous kernels to graph domains. These can recover most non-dynamic convolutional GNNs, from spatial to (polynomial) spectral. Addressing the fundamentally different characteristics of graph domains from Euclidean domains, we propose three theoretically and empirically motivated design innovations to accomplish the generalization to graphs. Theoretically, we demonstrate that CKGConv possesses equivalent expressive power to SOTA Graph Transformers in terms of distinguishing non-isomorphic graphs via the GD-WL test (Zhang et al., 2023). We also provide theoretical connections to previous convolutional GNNs. Empirically, the proposed CKGConv architecture either surpasses or achieves performance comparable to the SOTA across a wide range of graph datasets. It outperforms all other convolutional GNNs and achieves performance comparable to SOTA Graph Transformers. A further exploratory experiment suggests that CKGConv can learn non-dynamic sharpening kernels and extracts information complementary to that learned by the self-attention modules of Graph Transformers. This motivates a potential novel avenue of combining CKGConv and SAs in a single architecture. Furthermore, the success of CKGConv motivates the generalization of continuous kernel convolutions to other non-Euclidean geometric spaces based on pseudo-coordinate designs.

[7] According to a two-sided t-test at the 5% significance level.

[8] *Edge-detection* refers to detecting signal discontinuities in signal processing.

Impact Statement

This paper presents work whose goal is to advance the field of Geometric/Graph Deep Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgment

LM is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).
We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) [funding reference number 260250] and of the Fonds de recherche du Québec.
Cette recherche a été financée par le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG), [numéro de référence 260250] et par les Fonds de recherche du Québec.

References

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

Arnaiz-Rodríguez, A., Begga, A., Escolano, F., and Oliver, N. M. DiffWire: Inductive Graph Rewiring via the Lovász Bound. In *Proc. Learn. Graphs Conf.*, 2022.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer Normalization. In *Adv. Neural Inf. Process. Syst. Deep Learn. Symp.*, 2016.

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

Black, M., Wan, Z., Mishne, G., Nayyeri, A., and Wang, Y. Comparing Graph Transformers via Positional Encodings. *arXiv:2402.14202*, 2024.

Bo, D., Shi, C., Wang, L., and Liao, R. Specformer: Spectral Graph Neural Networks Meet Transformers. In *Proc. Int. Conf. Learn. Represent.*, 2023.

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

Bouritsas, G., Frasca, F., Zafeiriou, S. P., and Bronstein, M. Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022.

Bresson, X. and Laurent, T. Residual Gated Graph ConvNets. *arXiv:1711.07553*, 2018.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral Networks and Locally Connected Networks on Graphs. In *Proc. Int. Conf. Learn. Represent.*, 2014.

Chamberlain, B., Rowbottom, J., Gorinova, M. I., Bronstein, M., Webb, S., and Rossi, E. GRAND: Graph Neural Diffusion. In *Proc. Int. Conf. Mach. Learn.*, 2021.

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

Chien, E., Peng, J., Li, P., and Milenkovic, O. Adaptive Universal Generalized PageRank Graph Neural Network. In *Proc. Int. Conf. Learn. Represent.*, 2021.

Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017.

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

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Adv. Neural Inf. Process. Syst.*, 2016.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proc. Int. Conf. Learn. Represent.*, 2021.

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

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

Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Graph Neural Networks with Learnable Structural and Positional Representations. In *Proc. Int. Conf. Learn. Represent.*, 2022b.

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

Frasca, F., Rossi, E., Eynard, D., Chamberlain, B., Bronstein, M., and Monti, F. SIGN: Scalable Inception Graph Neural Networks. In *Proc. Int. Conf. Mach. Learn. Graph Represent. Learn. Beyond Workshop*, 2020.

Gabrielsson, R. B., Yurochkin, M., and Solomon, J. Rewiring with Positional Encodings for Graph Neural Networks. *Trans. Mach. Learn. Res.*, August 2023.

Gasteiger, J., Bojchevski, A., and Günnemann, S. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *Proc. Int. Conf. Learn. Represent.*, 2019a.

Gasteiger, J., Weißenberger, S., and Günnemann, S. Diffusion Improves Graph Learning. In *Adv. Neural Inf. Process. Syst.*, volume 32, 2019b.

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

Gutteridge, B., Dong, X., Bronstein, M., and Di Giovanni, F. DRew: Dynamically Rewired Message Passing with Delay. In *Proc. Int. Conf. Mach. Learn.*, 2023.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive Representation Learning on Large Graphs. In *Adv. Neural Inf. Process. Syst.*, volume 30, 2017.

Hammond, D. K., Vandergheynst, P., and Gribonval, R. Wavelets on Graphs Via Spectral Graph Theory. *Appl. Comput. Harmon. Anal.*, 30(2), March 2011.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Identity Mappings in Deep Residual Networks. In *Proc. Eur. Conf. Comput. Vis.*, volume 9908, 2016b.

He, M., Wei, Z., Huang, Z., and Xu, H. BernNet: Learning Arbitrary Graph Spectral Filters via Bernstein Approximation. In *Adv. Neural Inf. Process. Syst.*, 2021.

Hendrycks, D. and Gimpel, K. Gaussian Error Linear Units (GELUs). *arXiv:1606.08415*, 2023.

Hermosilla, P., Ritschel, T., Vázquez, P.-P., Vinacua, À., and Ropinski, T. Monte Carlo Convolution for Learning on Non-Uniformly Sampled Point Clouds. *ACM Trans. Graph.*, 37(6), December 2018.

Hornik, K., Stinchcombe, M., and White, H. Multilayer Feedforward Networks Are Universal Approximators. *Neural Netw.*, 2(5), January 1989.

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

Hua, B.-S., Tran, M.-K., and Yeung, S.-K. Pointwise Convolutional Neural Networks. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.

Hussain, M. S., Zaki, M. J., and Subramanian, D. Global Self-Attention as a Replacement for Graph Convolution. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2022.

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

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., and Coleman, R. G. ZINC: A Free Tool to Discover Chemistry for Biology. *J. Chem. Inf. Model.*, 52(7), July 2012.

Jia, X., De Brabandere, B., Tuytelaars, T., and Gool, L. V. Dynamic Filter Networks. In *Adv. Neural Inf. Process. Syst.*, volume 29, 2016.

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

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

Knigge, D. M., Romero, D. W., Gu, A., Gavves, E., Bekkers, E. J., Tomczak, J. M., Hoogendoorn, M., and Sonke, J.-j. Modelling Long Range Dependencies in ND: From Task-Specific to a General Purpose CNN. In *Proc. Int. Conf. Learn. Represent.*, 2023.

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

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In *Adv. Neural Inf. Process. Syst.*, volume 25, 2012.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *Proc. Int. Conf. Mach. Learn.*, 2019.

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

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

Liao, R., Zhao, Z., Urtasun, R., and Zemel, R. LanczosNet: Multi-Scale Deep Graph Convolutional Networks. In *Proc. Int. Conf. Learn. Represent.*, 2022.

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

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

Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., Wei, F., and Guo, B. Swin Transformer V2: Scaling Up Capacity and Resolution. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022a.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A ConvNet for the 2020s. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022b.

Loshchilov, I. and Hutter, F. SGDR: Stochastic Gradient Descent with Warm Restarts. In *Proc. Int. Conf. Learn. Represent.*, 2017.

Loshchilov, I. and Hutter, F. Decoupled Weight Decay Regularization. In *Proc. Int. Conf. Learn. Represent.*, 2019.

Loukas, A. What Graph Neural Networks Cannot Learn: Depth Vs Width. In *Proc. Int. Conf. Learn. Represent.*, 2020.

Luo, S., Li, S., Zheng, S., Liu, T.-Y., Wang, L., and He, D. Your Transformer May Not be as Powerful as You Expect. In *Adv. Neural Inf. Process. Syst.*, 2022.

Ma, L., Rabbany, R., and Romero-Soriano, A. Graph Attention Networks with Positional Embeddings. In *Proc. Pac. Asia Conf. Knowl. Discov. Data Min.*, 2021.

Ma, L., Lin, C., Lim, D., Romero-Soriano, A., K. Dokania, P., Coates, M., H.S. Torr, P., and Lim, S.-N. Graph Inductive Biases in Transformers without Message Passing. In *Proc. Int. Conf. Mach. Learn.*, 2023.

Mialon, G., Chen, D., Selosse, M., and Mairal, J. GraphiT: Encoding Graph Structure in Transformers. *arXiv:2106.05667*, 2021.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Proc. Eur. Conf. Comput. Vis.*, 2020.

Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., and Bronstein, M. M. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *Proc. AAAI Conf. Artif. Intell.*, volume 33, 2019.

Oono, K. and Suzuki, T. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *Proc. Int. Conf. Learn. Represent.*, 2020.

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

Park, W., Chang, W., Lee, D., Kim, J., and Hwang, S.-w. GRPE: Relative Positional Encoding for Graph Transformer. *arXiv:2201.12787*, March 2022.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017.

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

Romero, D. W., Kuzina, A., Bekkers, E. J., Tomczak, J. M., and Hoogendoorn, M. CKConv: Continuous Kernel Convolution For Sequential Data. In *Proc. Int. Conf. Learn. Represent.*, 2022.

Segol, N. and Lipman, Y. On Universal Equivariant Set Networks. In *Proc. Int. Conf. Learn. Represent.*, 2020.

Sitzmann, V., Martel, J. N. P., Bergman, A. W., Lindell, D. B., and Wetzstein, G. Implicit Neural Representations with Periodic Activation Functions. In *Adv. Neural Inf. Process. Syst.*, 2020.

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

Tan, M. and Le, Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proc. Int. Conf. Mach. Learn.*, 2019.

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *Adv. Neural Inf. Process. Syst.*, volume 33, 2020.

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., Lucic, M., and Dosovitskiy, A. MLP-Mixer: An all-MLP Architecture for Vision. In *Adv. Neural Inf. Process. Syst.*, 2021.

Tönshoff, J., Ritzert, M., Wolf, H., and Grohe, M. Walking Out of the Weisfeiler Leman Hierarchy: Graph Learning Beyond Message Passing. *Trans. Mach. Learn. Res.*, August 2023.

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

Touvron, H., Bojanowski, P., Caron, M., Cord, M., El-Nouby, A., Grave, E., Izacard, G., Joulin, A., Synnaeve, G., Verbeek, J., and Jégou, H. ResMLP: Feedforward Networks for Image Classification with Data-Efficient Training. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(4), April 2023.

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

Veličković, P. Message Passing All the Way Up. In *Proc. Int. Conf. Learn. Represent. Workshop Geometr. Topol. Represent. Learn.*, 2022.

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

Veličković, P., Ying, R., Padovano, M., Hadsell, R., and Blundell, C. Neural Execution of Graph Algorithms. In *Proc. Int. Conf. Learn. Represent.*, 2020.

Velingker, A., Sinop, A., Ktena, I., Veličković, P., and Gollapudi, S. Affinity-Aware Graph Networks. In *Adv. Neural Inf. Process. Syst.*, volume 36, 2023.

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

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., Lu, T., Luo, P., and Shao, L. Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions. In *Proc. IEEE Int. Conf. Comput. Vis.*, 2021.

Wang, X. and Zhang, M. How Powerful are Spectral Graph Neural Networks. In *Proc. Int. Conf. Mach. Learn.*, 2022.

Woo, S., Debnath, S., Hu, R., Chen, X., Liu, Z., Kweon, I. S., and Xie, S. ConvNeXt V2: Co-Designing and Scaling ConvNets With Masked Autoencoders. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023.

Wu, W., Qi, Z., and Fuxin, L. PointConv: Deep Convolutional Networks on 3D Point Clouds. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.

Xiao, T., Singh, M., Mintun, E., Darrell, T., Dollar, P., and Girshick, R. Early Convolutions Help Transformers See Better. In *Adv. Neural Inf. Process. Syst.*, volume 34, 2021.

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

Xu, Y., Fan, T., Xu, M., Zeng, L., and Qiao, Y. SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters. In *Proc. Eur. Conf. Comput. Vis.*, 2018.

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

You, J., Ying, R., and Leskovec, J. Position-aware Graph Neural Networks. In *Proc. Int. Conf. Mach. Learn.*, 2019.

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

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep Sets. In *Adv. Neural Inf. Process. Syst.*, 2017.

Zhang, B., Luo, S., Wang, L., and He, D. Rethinking the Expressive Power of GNNs via Graph Biconnectivity. In *Proc. Int. Conf. Learn. Represent.*, 2023.

Zhang, Z., Cui, P., Pei, J., Wang, X., and Zhu, W. Eigen-GNN: A Graph Structure Preserving Plug-in for GNNs. *IEEE Trans. Knowl. Data Eng.*, 2021.

Zhao, J., Dong, Y., Ding, M., Kharlamov, E., and Tang, J. Adaptive Diffusion in Graph Neural Networks. In *Adv. Neural Inf. Process. Syst.*, volume 34, 2021.

Zhao, L., Jin, W., Akoglu, L., and Shah, N. From Stars to Subgraphs: Uplifting Any GNN with Local Structure Awareness. In *Proc. Int. Conf. Learn. Represent.*, 2022.

Zhou, C., Wang, X., and Zhang, M. Facilitating Graph Neural Networks with Random Walk on Simplicial Complexes. In *Adv. Neural Inf. Process. Syst.*, volume 36, 2023.

A. Model Architecture and Implementation Details

A.1. Model Architecture

In order to combine all the building blocks into one clear visualization, we provide an overview of the CKGCN in Figure 3.

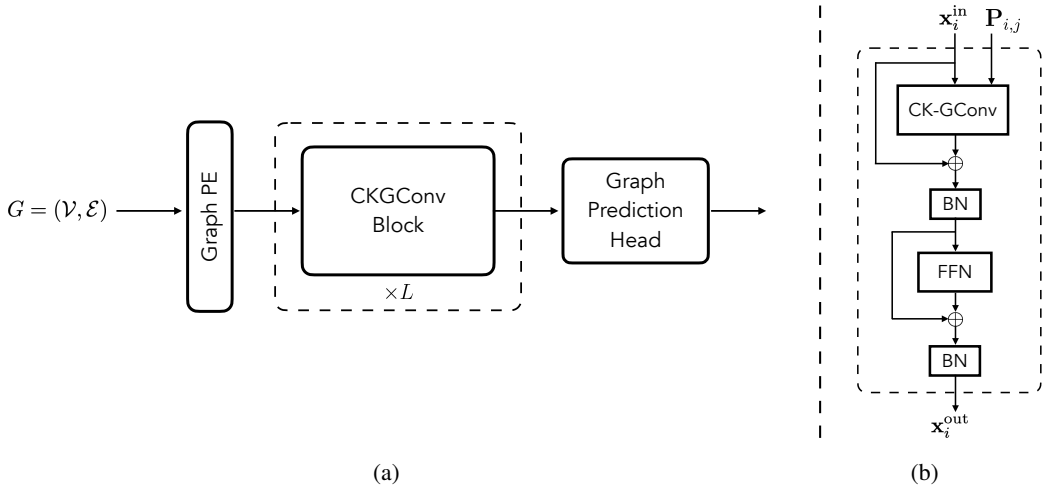

Figure 3. (a) Detailed Architecture of CKGCN with L CKGConv blocks and task-dependent output head, (b) the detailed design of each CKGConv-block.

A.2. Rescaling of RRWP

The expected values of random walk based graph PEs, e.g., RRWP, are dependent on the graph orders. For a graph with N nodes, RRWP has the property that $\sum_{j \in \mathcal{V}} \mathbf{P}_{i,j} = \mathbf{1} \Rightarrow \mathbb{E}_{j \sim \mathcal{V}}[\mathbf{P}_{i,j}] = \mathbf{1}/N$. Empirically, we found that removing this dependency is beneficial to CKGConv. Therefore, we introduce an extra re-scaling for RRWP by setting $\mathbf{P}_{i,j} \leftarrow N \cdot \mathbf{P}_{i,j}$. For other graph PEs without such dependencies, e.g,. RD and SPD, this re-scaling is not necessary.

Following the approaches in GraphGPS (Rampášek et al., 2022), we introduce an extra BatchNorm (Ioffe & Szegedy, 2015) on the input RRWP to further normalize the input values.

B. Additional Related Work

Graph Positional Encoding In recent years, positional and/or structural encoding has been widely studied to enhance the performance of MPNNs (You et al., 2019; Ma et al., 2021; Li et al., 2020; Zhang et al., 2021; Loukas, 2020; Dwivedi et al., 2022b; Lim et al., 2023; Wang et al., 2022; You et al., 2021; Velingker et al., 2023; Bouritsas et al., 2022). Due to the inherent properties of attention mechanisms (Vaswani et al., 2017; Lee et al., 2019), Graph Transformers rely on positional/structural encoding even more excessively. Disparate designs have been proposed by previous works, from absolute ones (Dwivedi & Bresson, 2021; Kreuzer et al., 2021; Kim et al., 2022) to relative ones (Ying et al., 2021; Zhang et al., 2023; Ma et al., 2023; Mialon et al., 2021; Hussain et al., 2022; Park et al., 2022). A recent work (Zhou et al., 2023) has also explored the potential of computing positional encoding on higher-order simplicial complexes instead of on nodes. Positional encodings prioritize distance/affinity measures and structural encodings focus on structural patterns, but most encodings incorporate both positional and structural information (Srinivasan & Ribeiro, 2020).

C. Experimental Details

C.1. Description of Datasets

Table 6 provides a summary of the statistics and characteristics of datasets used in this paper. The first five datasets are from Dwivedi et al. (2022a), and the last two are from Dwivedi et al. (2022c). Readers are referred to Rampášek et al. (2022) for more details about the datasets.

Table 6. Overview of the graph learning datasets involved in this work (Dwivedi et al., 2022a;c; Irwin et al., 2012).

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

C.2. Dataset splits and random seed

We conduct the experiments on the standard train/validation/test splits of the evaluated benchmarks, following previous works (Rampášek et al., 2022; Ma et al., 2023). For each dataset, we execute 4 runs with different random seeds (0,1,2,3) and report the mean performance and standard deviation.

C.3. Optimizer and Learning Rate Scheduler

We use AdamW (Loshchilov & Hutter, 2019) as the optimizer and the Cosine Annealing Learning Rate scheduler (Loshchilov & Hutter, 2017) with linear warm up.

C.4. Hyperparameters

Due to the limited time and computational resources, we did not perform an exhaustive search or a grid search for the hyperparameters. We mostly follow the hyperparameter settings of GRIT (Ma et al., 2023), and make slight changes to adjust the number of parameters to match the commonly used parameter budgets.

We follow the most commonly used parameter budgets: up to 500k parameters for ZINC, PATTERN, CLUSTER, Peptides-func and Peptides-struct; and around 100k parameters for MNIST and CIFAR10.

The final hyperparameters are presented in Table 7 and Table 8.

C.5. Significance Test

We conduct a two-sample one-tailed t-test to verify the statistical significance of the difference in performance. The baselines' results are taken from (Ma et al., 2023).

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

C.6. Runtime

We provide the runtime and GPU memory consumption of CKGCN in comparison to GRIT on ZINC as reference (Table 9). The timing is conducted on a single NVIDIA V100 GPU (Cuda 11.8) and 20 threads of Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz.

C.7. Efficient Implementation

A more efficient implementation is achievable for CKGConv with global support when the graph order is large. Based on the following derivation, we can implement an algorithm with $O(|\mathcal{V}|S)$ complexity, where $S := \mathbb{E}_{i\sim\mathcal{V}}[|\{\mathbf{P}_{i,j} \neq \mathbf{0}\}|]$. The complexity thus depends on the order of the RRWP and the graph structure. .

Table 7. Hyperparameters for five datasets from BenchmarkingGNNs (Dwivedi et al., 2022a).

Hyperparameter	ZINC	MNIST	CIFAR10	PATTERN	CLUSTER
# CKGConv-Block	10	4	3	10	16
- Hidden dim	64	48	56	64	54
- Dropout	0	0	0	0	0.01
- Norm.	BN	BN	BN	BN	BN
Graph pooling	sum	mean	mean	–	–
PE dim (K-RRWP)	21	18	18	21	32
Kernel Func.					
- # MLP Block	2	2	2	2	2
- Norm.	BN	BN	BN	BN	BN
- Kernel dropout	0.5	0.5	0.5	0.5	0.5
- MLP dropout	0.1	0.2	0.	0.2	0.5
Batch size	32	16	16	16	16
Learning Rate	0.001	0.001	0.001	0.001	0.001
# Epochs	2000	200	200	200	200
# Warmup epochs	50	5	5	10	10
Weight decay	$1e-5$	$1e-5$	$1e-5$	$1e-5$	$1e-5$
Min. lr.	$1e-6$	$1e-4$	$1e-4$	$1e-4$	$1e-4$
# Parameters	433,663	102,580	105,320	438,143	499,754

Let $\mathcal{S}_i := \{j \in \mathcal{V} : \mathbf{P}_{i,j} \neq \mathbf{0}\}$, then ignoring the bias term, Eq. (3) can be written as

$$(\chi \star \psi)(i) = \frac{1}{|\mathcal{V}|}\left(\sum_{j\in\mathcal{S}_i} \chi(j)|\mathcal{V}| \cdot \psi(\mathbf{P}_{i,j}) + \sum_{j'\in\mathcal{V}\setminus\mathcal{S}_i} \chi(j)\cdot\psi(\mathbf{0})\right), \tag{11}$$

$$= \frac{1}{|\mathcal{V}|}\left(\sum_{j\in\mathcal{S}_i} \chi(j)|\mathcal{V}| \cdot \big(\psi(\mathbf{P}_{i,j}) - \psi(\mathbf{0})\big)\right) + \frac{1}{|\mathcal{V}|}\left(\sum_{j\in\mathcal{V}} \chi(j)\cdot\psi(\mathbf{0})\right), \tag{12}$$

$$= \frac{1}{|\mathcal{V}|}\left(\sum_{j\in\mathcal{S}_i} \chi(j)|\mathcal{V}| \cdot \big(\psi(\mathbf{P}_{i,j}) - \psi(\mathbf{0})\big)\right) + \psi(\mathbf{0})\cdot\frac{1}{|\mathcal{V}|}\left(\sum_{j\in\mathcal{V}} \chi(j)\right). \tag{13}$$

The second term of Eq. (13) can be computed by global-average pooling of graphs shared by all nodes in $O(|\mathcal{V}|)$, and the first term requires $O(|\mathcal{V}| \cdot S)$ computation on average, where $S = \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} |\mathcal{S}_i|$.

D. Additional Experiments: Toy Examples, Sensitivity Study, and Ablation Study

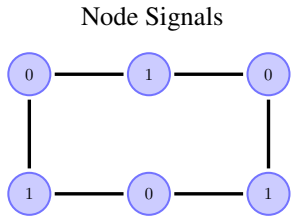

Figure 4. The toy example for *Anti-oversmoothing*.

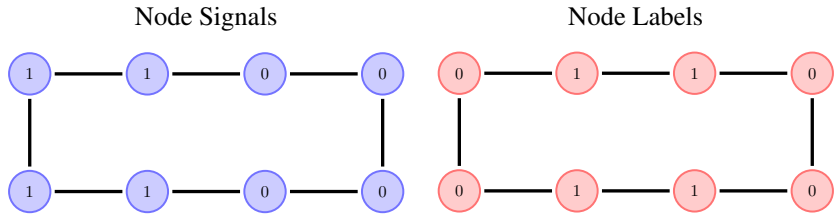

Figure 5. The toy example for *Edge Detection*.

Table 8. Hyperparameters for two datasets from the Long-range Graph Benchmark (Dwivedi et al., 2022c).

Hyperparameter	Peptides-func	Peptides-struct
# CKGConv-Block	4	4
- Hidden dim	96	96
- Dropout	0	0.05
- Norm.	BN	BN
Graph pooling	mean	mean
PE dim (K-RRWP)	24	24
Kernel Func.		
- # MLP Block	2	2
- Norm.	BN	BN
- Kernel dropout	0.5	0.2
- MLP dropout	0.2	0.2
Batch size	16	16
Learning Rate	0.001	0.001
# Epochs	200	200
# Warmup epochs	5	5
Weight decay	0	0
Min. lr.	1e-4	1e-4
# Parameters	421,468	412,253

D.1. Toy Example: CKGConv Can Mitigate Oversmoothing

With the ability to generate both positive and negative coefficients, CKGConv can learn sharpening kernels (a.k.a. high-pass filters), which amplify the signal differences among data points to alleviate oversmoothing. Here, we provide a toy example to better illustrate CKGConv's capability to prevent oversmoothing.

We consider a simple graph with node signals as shown in Fig. 4, and train 2-layer and 6-layer GCNs and CKGCNs, with 5-RRWP, to predict labels that match the node signals. In this toy example, we remove all normalization layers, dropouts, as well as residual connections. All models are trained for 200 epochs with the Adam optimizer (initial learning rate 1e-3) to overfit this binary classification task. We report the results of 5 trials with different random seeds in Table 10.

As shown in the results, both the 2-layer GCN and 2-layer CKGCN can overfit the toy example and reach 100% accuracy. However, a 6-layer GCN fails to reconstruct the node signals. Applying 6 smoothing convolutions (all-positive filter coefficients) in this small network leads to the aggregated representation at each node being very similar. This is a typical oversmoothing effect. The network predicts all nodes to have the same label, resulting in 50% accuracy in the toy example. In contrast, a 6-layer CKGCN not only reaches 100% accuracy but also achieves a lower BCE loss, showcasing its strong capability in mitigating oversmoothing.

Table 9. Runtime and GPU memory for GRIT (Ma et al., 2023) and CKGCN (Ours) on ZINC with batch size 32. The timing is conducted on a single NVIDIA V100 GPU (Cuda 11.8) and 20 threads of Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz.

ZINC	CKGConv	GRIT
GPU Memory	2146 MB	1896 MB
Training time	35.9 sec/epoch	39.7 sec/epoch

Table 10. Toy Example for Anti-oversmoothing (Fig. 4): Training performance for reconstruction of node signals. Shown is the mean ± s.d. of 5 runs with different random seeds.

Train	2-Layer GCN	6-Layer GCN	2-Layer CKGCN	6-Layer CKGCN
BCE Loss	0.071 ± 0.044	0.693 ± 2e-05	4e-05± 2e-05	0.0 ± 0.0
Accuracy (%)	100 ± 0	50 ± 0	100 ± 0	100 ± 0

D.2. Toy Example: CKGConv Can Do Edge-Detection

Analogous to *edge-detection* in signal processing, in the graph domain, kernels with positive and negative coefficients can be used to detect the nodes with cross-community-connection (the border nodes). In such a setting, it is essential that the sign of the filter coefficient for the central node is opposite to those of the first-hop neighbors, in order to detect differences in attributes.

We introduce a toy example to demonstrate it as shown in Fig. 5: given a graph with simple scalar node signals that match the "community" of a node (0 or 1), the goal is to identify the border nodes with node labels as 1,

In this study, we consider single-channel convolution kernels. We compare CKGConv with three all-positive kernels: GCNConv (Kipf & Welling, 2017), CKGConv+Softmax (attention-like), and CKGConv+Softplus. CKGConv and its variants use 5-RRWP with the hidden dimension of 5 in the kernel function. CKGCN+Softmax (sum-aggregation) and CKGCN+Softplus (mean-aggregation) apply Softmax and Softplus on the kernel coefficients, respectively, to constrain the kernels to have positive coefficients only.

We aim to verify the upper bounds for the expressivity of the convolutions by training them to overfit the task. Each convolution operator is trained for 200 epochs with the Adam optimizer (learning rate 1e-2) using binary cross entropy loss (BCE loss). We report the last training BCE loss and accuracy from 5 trials with different initializations, in Table 11.

Table 11. Toy Example for Edge-detection (Fig. 5): Training performance for reconstruction of node signals. Shown is the mean ± s.d. of 5 runs with different random seeds.

Train	CKGConv	GCNConv	CKGConv+Softmax	CKGConv+Softplus
BCE Loss	**2e-4 ± 1e-05**	0.693 ± 0.001	0.693 ± 0	0.687 ± 0.049
Accuracy (%)	**100 ± 0**	50 ± 0	50 ± 0	60 ± 12.25

From the results, it is obvious that only convolution kernels with negative and positive values (regular CKConv) can reach 100% accuracy and achieve a low BCE loss. All other convolution kernels, with only positive values, fail to identify the border nodes.

This toy example explains why negative coefficients are advantageous in graph learning tasks, which require the detection of signal differences among data points. Similar tasks include *ridge-detection* and *learning on heterophilic graphs*.

D.3. Sensitivity study on kernel support sizes of CKGConv

The CKGConv framework allows for kernels with pre-determined non-global supports, analogous to the regular convolution in Euclidean spaces. In this section, we study the effect of different pre-determined support sizes based on K-hop neighborhoods on ZINC datasets. The sensitivity study follows the same experimental setup as the main experiments. The timing is conducted on a single a single NVIDIA V100 GPU (Cuda 11.8) and 20 threads of Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz.

From the results in Table 12, with the same order of RRWP, larger support sizes usually lead to better empirical performances as well as greater GPU memory consumption. On the one hand, the results showcase the stability in performance of CKGConv, since all CKGCN variants with $K>1$ hops reach competitive performance among the existing graph models, outperforming all existing GNNs and most Graph Transformers. On the other hand, the results justify the necessity of introducing graph convolutions beyond the one-hop neighborhood (a.k.a., message-passing), since the one-hop CKGCN

Table 12. The sensitivity study of the support sizes (K-hop neighborhoods) for CKGConv kernels with 21-RRWP.

ZINC	MAE ↓	Run-Time (sec/epoch)	GPU-Mem (MB)
1-hop	0.073 ± 0.005	33.1	1186
3-hop	0.063 ± 0.002	33.6	1522
5-hop	0.061 ± 0.004	35.2	1624
11-hop	0.063 ± 0.002	33.2	2128
21-hop	0.060 ± 0.002	34.8	2148
Full	0.059 ± 0.003	35.4	2148

is significantly worse than the other variants with larger kernels. Furthermore, the sensitivity study also highlights the flexibility of CKGConv framework in balancing the computational cost and the capacity to model long-range dependencies, by effortlessly controlling the kernel sizes like the Euclidean convolutions. Note that the kernel sizes are not necessarily tied with the order of RRWP or the counterparts of other graph PEs in CKGConv.

D.4. Ablation study on the kernel functions.

As depicted in Sec. 4.3, polynomial-based GNNs can be viewed as CKGCN with linear kernel functions. However, allowing kernel functions with non-linearity is important, since multilayer perceptions (MLPs) with non-linear activations can be universal function approximators (Hornik et al., 1989) while linear functions cannot.

To better understand the effects of the choices of kernel functions, we conduct an ablation study on ZINC and PATTERN datasets, following the experimental setup of the main experiments. We compare different CKGCN variants, using kernel functions with $0, 1,$ and 2 MLP-blocks (as shown in Eq. (5)). The width of each variant is adjusted to reach the parameter budget under 500 K. Note that 0 MLP-blocks is equivalent to a linear kernel function and 2 MLP-blocks setting is the default in CKGCN.

Table 13. The ablation study on # MLP blocks in the kernel function of CKGConv.

# MLP	ZINC		PATTERN	
Blocks	MAE ↓	# param.	W.Accuracy ↑	# param.
0	0.074 ± 0.005	487 K	87.355 ± 0.230	495 K
1	0.065 ± 0.005	438 K	88.955 ± 0.251	444 K
2	0.059 ± 0.003	434 K	88.661 ± 0.142	438 K

From the results of the ablation study (as shown in Table 13), CKGCNs with linear kernel functions under-perform the variants with non-linear kernel functions on both ZINC and PATTERN datasets, even with more learnable parameters. This observation matches our hypothesis on the indispensability of the non-linearity in kernel functions. It also justifies the advantage of CKGConv framework over the previous polynomial GNNs which can only introduce linear kernel functions.

E. Theory and Proof

E.1. The Expressiveness of CKGConv Is Equivalent to GD-WL

We use a Weisfeiler-Lehman (WL)-like graph isomorphism framework to analyze theoretical expressiveness. Specifically, we consider the *Generalized Distance WL* (GD-WL) test, which is based on updating node colors incorporating graph distances proposed by Zhang et al. (2023).

For a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the iterative node color update in GD-WL test is defined as:

$$\chi^{\ell}_{\mathcal{G}}(v) = \mathit{hash}(\{\!\{(d_{\mathcal{G}}(v,u), \chi^{\ell-1}_{\mathcal{G}}(u)) : u \in \mathcal{V}\}\!\})\,. \tag{14}$$

where $d_{\mathcal{G}}(v,u)$ denotes a distance between nodes v and u, and $\chi^{0}_{G}(v)$ is the initial color of v. The multiset of final node

colors $\{\!\{\chi_G^L(v) : v \in \mathcal{V}\}\!\}$ at iteration L is hashed to obtain a graph color.

Our proof for the expressiveness of CKGConv employs the following lemma provided by Xu et al. (2019).

Lemma E.1. *(Lemma 5 of Xu et al. (2019)) For any countable set $\mathcal{X}$, there exists a function $f : \mathcal{X} \to \mathbb{R}^n$ such that $h(\hat{\mathcal{X}}) := \sum_{x \in \hat{\mathcal{X}}} f(x)$ is unique for each multiset $\hat{\mathcal{X}} \in \mathcal{X}$ of bounded size. Moreover, for some function ϕ, any multiset function g can be decomposed as $g(\hat{\mathcal{X}}) = \phi(\sum_{x \in \hat{\mathcal{X}}} f(x))$.*

Proof of Proposition 3.1. In this proof, we consider shortest-path distance (SPD) as an example of generalized distance (GD). This is denoted as $\mathrm{d}_G^{\mathrm{SPD}}$ and is assumed to construct the pseudo-coordinates in CKGConv. The proof holds with other GDs such as the resistance distance (RD) (Zhang et al., 2023) and RRWP (Ma et al., 2023), and the choice of GD determines the practical expressiveness of GD-WL.

We consider all graphs with at most n nodes to distinguish in the isomorphism tests. The total number of possible values of d_G is finite and depends on n (upper bounded by n^2). We define

$$\mathcal{D}_n = \{\mathrm{d}_G^{\mathrm{SPD}}(u, v) : G = (\mathcal{V}, \mathcal{E}), |\mathcal{V}| \leqslant n, u, v \in \mathcal{V}\}\,, \tag{15}$$

to denote all possible values of $\mathrm{d}_G^{\mathrm{SPD}}(u, v)$ for any graphs with at most n nodes. We note that since $\mathcal{D}_n$ is a finite set, its elements can be listed as $\mathcal{D}_n = \{d_{G,1}, \cdots, d_{G,|\mathcal{D}_n|}\}$.

Then the GD-WL aggregation at the ℓ-th iteration in Eq. (14) can be equivalently rewritten as (See Theorem E.3 in Zhang et al. (2023)):

$$\begin{aligned} \chi_G^{\ell}(v) &:= \mathrm{hash}\Big(\chi_G^{\ell,1}(v), \chi_G^{\ell,2}(v), \cdots, \chi_G^{\ell,|\mathcal{D}_n|}(v)\Big)\,, \\ \text{where } \chi_G^{\ell,k}(v) &:= \{\!\{\chi_G^{\ell-1}(u) : u \in \mathcal{V}, \mathrm{d}_G(u, v) = d_{G,k}\}\!\}\,. \end{aligned} \tag{16}$$

In other words, for each node v, we can perform a color update by hashing a tuple of color multisets. We construct the k-th multiset by injectively aggregating the colors of all nodes $u \in \mathcal{V}$ at a distance $d_{G,k}$ from node v.

Assuming the color of each node $\chi_G^t(v)$ is represented as a vector $\mathbf{x}_v^{(l)} \in \mathbb{R}^C$, and setting the bias $\mathbf{b}$ to $\mathbf{0}$ for simplicity, the l-th CK-GConv layer with a global support (as shown in Eq. (4)) can be written as

$$\hat{\mathbf{x}}_v^{(l)} := \frac{1}{|\mathcal{V}|} \sum_{u \in \mathcal{V}} (\mathbf{W}\mathbf{x}_u^{(l)}) \odot \boldsymbol{\psi}\big(\mathrm{d}_G(u, v)\big)\,. \tag{17}$$

where $\boldsymbol{\psi} : \mathbb{R} \to \mathbb{R}^C$ and $\mathbf{W} \in \mathbb{R}^{C \times C}$ is the learnable weight. Then, we will show that with certain choices of the kernel function, a CKGCN is as powerful as GD-WL.

First, we define the kernel function $\boldsymbol{\psi}$ as a composition of H sub-kernel functions $\{\boldsymbol{\psi}^h : \mathbb{R} \to \mathbb{R}^F\}_{h=1,\ldots,H}$ such that $\boldsymbol{\psi}(d) = [\boldsymbol{\psi}^1(d) \| \ldots \| \boldsymbol{\psi}^H(d)] \in \mathbb{R}^C, \forall d \in \mathcal{D}_n$, where $[\cdot\|\cdot]$ denotes the concatenation of vectors and $C = H \cdot F$.

Then Eq. (17) can be written as

$$\hat{\mathbf{x}}_v^{(l),h} := \frac{1}{|\mathcal{V}|} \sum_{u \in \mathcal{V}} (\mathbf{W}^h \mathbf{x}_u^{(l)}) \odot \boldsymbol{\psi}^h\big(\mathrm{d}_G(u, v)\big)\,, \tag{18}$$

$$\hat{\mathbf{x}}_v^{(l)} = [\hat{\mathbf{x}}_v^{(l),1} \| \cdots \| \hat{\mathbf{x}}_v^{(l),H}]\,, \tag{19}$$

where $\mathbf{W} \in \mathbb{R}^{C \times C}$ is partitioned as $[\mathbf{W}^{1\intercal}, \cdots, \mathbf{W}^{H\intercal}]^\intercal$ so that each $\mathbf{W}^h \in \mathbb{R}^{F \times C}$.

We construct $\boldsymbol{\psi}^h(d) := \mathbb{I}(d = d_{G,h}) \cdot \mathbf{1}$, where $\mathbb{I} : \mathbb{R} \to \mathbb{R}$ is the indicator function, $d_{G,h} \in \mathcal{D}_n$ is a pre-determined

condition, and $\mathbf{1} \in \mathbb{R}^F$. Then, the convolution by each sub-kernel (Eq. (18) can be written as

$$\begin{aligned}
\hat{\mathbf{x}}_v^{(l),h} :=& \frac{1}{|\mathcal{V}|} \sum_{u \in \mathcal{V}} (\mathbf{W}^h \mathbf{x}_u^{(l)}) \odot \boldsymbol{\psi}^h \big(\mathrm{d}_G(u,v)\big)\,, \\
=& \frac{1}{|\mathcal{V}|} \sum_{u \in \mathcal{V}} (\mathbf{W}^h \mathbf{x}_u^{(l)}) \odot \big(\mathbb{I}(\mathrm{d}_G(u,v) = d_{G,h}) \cdot \mathbf{1}\big)\,, \\
=& \frac{1}{|\mathcal{V}|} \sum_{u \in \mathcal{V}} (\mathbf{W}^h \mathbf{x}_u^{(l)}) \cdot \mathbb{I}(\mathrm{d}_G(u,v) = d_{G,h})\,, \\
=& \frac{1}{|\mathcal{V}|} \sum_{\mathrm{d}_G(u,v) = d_{G,h}} \mathbf{W}^h \mathbf{x}_u^{(l)}\,.
\end{aligned} \tag{20}$$

Note that $\mathbf{W}$ can be absorbed as the last layer of the feed-forward network (FFN) in the previous layer. Because $\mathbf{x}_u^{(l)}$ is processed by the FFN in the previous layer, we can invoke Lemma E.1 to establish that each sub-kernel $\boldsymbol{\psi}^h$ (as in Eq. (20)) can implement an injective aggregating function for $\{\!\{\chi_G^{t-1}(u) : u \in \mathcal{V}, d_G(u,v) = d_{G,h}\}\!\}$. The concatenation in Eq. (19) is an injective mapping of the tuple of multisets $\left(\chi_G^{t,1}, \cdots, \chi_G^{t,|\mathcal{D}_n|}\right)$. When any of the linear mappings has irrational weights, the projection will also be injective. Therefore, one CKGConv followed by the FFN can implement the aggregation formula (Eq. (16)), with a sufficiently large number of different $\boldsymbol{\psi}^h$. Thus, the CKGCN can perform the aggregation of GD-WL. Therefore, with a sufficiently large number of layers, CKGCN is as powerful as GD-WL in distinguishing non-isomorphic graphs, which concludes the proof. □

E.2. CKGConv and Equivariant Set Neural Networks

Proof of Proposition 4.1. We prove the proposition for scalar-valued signals, which can be directly generalized to vector-valued signals.

For a globally supported CKGConv, given the 1-RRWP after the re-scaling (Appendix A.2) $P_{i,i} = |\mathcal{V}|$ and $P_{i,j} = 0, \forall i,j \in \mathcal{V}, i \neq j$, denoted as P_0 and P_1 for simplicity. Considering $\psi : \mathbb{R} \to \mathbb{R}$ that $\psi(x) = \gamma \cdot x + \beta\ \gamma, \beta \in \mathbb{R}$, Eq. (3) can be written as

$$\begin{aligned}
(\chi \star \psi)(i) &= \frac{1}{|\mathcal{V}|} \left(\chi(i)(|\mathcal{V}| \cdot \gamma + \beta) + \sum_{j \in \mathcal{V}; j \neq i} \chi(j)\beta \right) + b\,, \\
&= \frac{1}{|\mathcal{V}|} \chi(i)(|\mathcal{V}| \cdot \gamma + \beta - \beta) + \frac{1}{|\mathcal{V}|} \sum_{j \in \mathcal{V}} \chi(j)\beta + b\,, \\
&= \gamma \cdot \chi(i) + \beta \cdot \big(\frac{1}{|\mathcal{V}|} \sum_{j \in \mathcal{V}} \chi(j)\big) + b\,.
\end{aligned} \tag{21}$$

This is the general form of a layer in an *Equivariant Set Network* (Eq. 8 in Segol & Lipman (2020)). This general form can cover a wide range of set neural networks (Zaheer et al., 2017; Qi et al., 2017). □

E.3. CKGConv, Polynomial Spectral GNNs and Diffusion Enhaned GNNs

Lemma E.2. *Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ denotes the adjacency matrix of an undirected graph G and the diagonal matrix $\mathbf{D} \in \mathbb{R}^{n \times n}, [\mathbf{D}]_{i,i} = \sum_{j \in \mathcal{V}} [\mathbf{A}]_{i,j}$ is the degree matrix, the k-power of symmetric normalized adjacency matrix $\tilde{\mathbf{A}} := \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ and random walk matrix $\mathbf{M} := \mathbf{D}^{-1}\mathbf{A}$, satisfy that*

$$\tilde{\mathbf{A}}^k = \mathbf{D}^{1/2}\mathbf{M}\mathbf{D}^{-1/2}, \forall k = 1, 2, \cdots \tag{22}$$

Proof of Lemma E.2. For arbitrary $k \geq 1$, we have

$$\begin{aligned}
\tilde{\mathbf{A}}^k &= (\mathbf{D}^{1/2}\mathbf{M}\mathbf{D}^{-1/2})^k \\
&= (\mathbf{D}^{1/2}\mathbf{M}\mathbf{D}^{-1/2})(\mathbf{D}^{1/2}\mathbf{M}\mathbf{D}^{-1/2}) \cdots (\mathbf{D}^{1/2}\mathbf{M}\mathbf{D}^{-1/2})\ ,k \text{ times} \\
&= \mathbf{D}^{1/2}\mathbf{M} \cdots \mathbf{M}\mathbf{D}^{-1/2} \\
&= \mathbf{D}^{1/2}\mathbf{M}^k\mathbf{D}^{-1/2}\,.
\end{aligned} \tag{23}$$

□

Proof of Proposition 4.2. Irrespective of the specific polynomial parameterization that is employed, any $K-1$ order *Polynomial Spectral Graph Neural Network* can be defined in a general form with $\tilde{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \in \mathbb{R}^{n\times n}$, parameterized by a learnable vector $\theta = [\theta_0, \cdots, \theta_{K-1}]^\intercal \in \mathbb{R}^{K\times 1}$ for the filtering of an input graph signal $\mathbf{x} \in \mathbb{R}^{n\times 1}$ to obtain an output graph signal $\mathbf{y} \in \mathbb{R}^{n\times 1}$, as follows:

$$\begin{aligned}
\mathbf{y} &= g_\theta(\tilde{\mathbf{L}})\mathbf{x}\,, \\
&= \sum_{k=0}^{K-1} \theta_k \tilde{\mathbf{L}}^k \mathbf{x}\,, \\
&= \sum_{k=0}^{K-1} \theta_k \sum_{r=0}^{k} \binom{k}{r} (-1)^r \tilde{\mathbf{A}}^r \mathbf{x}\,, \\
&= \sum_{k=0}^{K-1} \theta'_k \tilde{\mathbf{A}}^k \mathbf{x}\,.
\end{aligned} \tag{24}$$

Here $\tilde{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ and $\theta'_k = \sum_{r=k}^{K-1} \binom{r}{k}(-1)^k \theta_r$. Therefore, the spectral filter $g_\theta(\tilde{\mathbf{L}})$ can be represented by a linear combination of a collection of polynomial bases $\{\mathbf{I}, \tilde{\mathbf{A}}^1, \tilde{\mathbf{A}}^2, \cdots, \tilde{\mathbf{A}}^{K-1}\}$. In other words,

$$\begin{aligned}
[g_\theta(\tilde{\mathbf{L}})]_{i,j} &= \psi([\mathbf{I}, \tilde{\mathbf{A}}^1, \tilde{\mathbf{A}}^2, \cdots, \tilde{\mathbf{A}}^{K-1}]_{i,j})\,, \\
&= \psi(d_i^{1/2}[\mathbf{I}, \mathbf{M}^1, \mathbf{M}^2, \cdots, \mathbf{M}^{K-1}]_{i,j} d_j^{-1/2})\,, \text{ using Lemma E.2} \\
&= d_i^{1/2}\psi([\mathbf{I}, \mathbf{M}^1, \mathbf{M}^2, \cdots, \mathbf{M}^{K-1}]_{i,j}) d_j^{-1/2}\,, \text{ as } \psi \text{ is a linear projection} \\
&= d_i^{1/2}\psi(\mathbf{P}_{i,j}) d_j^{-1/2}\,, \\
&= \frac{1}{S} \cdot d_i^{1/2}\psi(S \cdot \mathbf{P}_{i,j}) d_j^{-1/2}\,.
\end{aligned} \tag{25}$$

Here $\psi : \mathbb{R}^K \to \mathbb{R}$ is a linear projection; $d_i = \mathbf{D}_{i,i} \in \mathbb{R}$ is the degree of node i; and $S \in \mathbb{R}$ is the scaling term in the scaled-convolution design and the RRWP rescaling.

In other words, with the K-RRWP as pseudo-coordinates, CKGConv with a linear kernel ψ can recover most *polynomial spectral GNNs* in the form of Eq. (24) irrespective of the specific polynomial parameterization that is used, if $d_i^{1/2}$ and $d_j^{-1/2}$ are injected properly, for all $i, j \in \mathcal{V}$. The result trivially holds for other Laplacian normalizations (e.g., row-normalized, max-eigenvalue normalized), where different constant multipliers are injected via adaptive degree scalers to $\mathbf{P}_{i,j}$.

Similarly, *Polynomial Diffusion Enhanced Graph Neural Networks* employing polynomials of $\tilde{\mathbf{A}}$ or its variants also can be represented by Eq. (24). Hence, the result follows. □

E.4. Degree Information and Normalization Layers

Normalization layers are essential for deep neural networks. Ma et al. (2023) provide a thorough discussion on the impact of normalization layers on the explicit injected degree information via sum-aggregation or degree scalers, which motivates our choice of BatchNorm (Ioffe & Szegedy, 2015) over LayerNorm (Ba et al., 2016).

Proposition E.3. *(Ma et al., 2023) Sum-aggregated node representations, degree-scaled node representations, and mean-aggregated node representations all have the same value after the application of a LayerNorm on node representations.*

Proof of Proposition E.3. Regardless the linear transformation in MPNN shared by nodes, we can write the output representation for a node i from a sum-aggregator as $\mathbf{x}_i^{\text{sum}} = d_i \cdot \mathbf{x}_i^{\text{mean}}$, where $d_i \in \mathbb{R}$ is the degree of node i and $\mathbf{x}_i^{\text{mean}} = [x_{i1}, \ldots x_{iF}]^\top \in \mathbb{R}^F$ is the node representation from a mean-aggregator. The layer normalization statistics

for a node i over all hidden units are computed as follows:

$$\begin{aligned} \mu_i^{\text{sum}} &= \frac{1}{F}\sum_{j=1}^{F} x_{ij}^{\text{sum}} = \frac{1}{F}\sum_{j=1}^{F} d_i \cdot x_{ij}^{\text{mean}} = \frac{d_i}{F}\sum_{j=1}^{F} x_{ij}^{\text{mean}} = d_i \cdot \mu_i^{\text{mean}} \\ \sigma_i^{\text{sum}} &= \sqrt{\frac{1}{F}\sum_{j=1}^{F}(x_{ij}^{\text{sum}} - \mu^{\text{sum}})^2} = \sqrt{\frac{d_i^2}{F}\sum_{j=1}^{F}(x_{ij}^{\text{mean}} - \mu^{\text{mean}})^2} = d_i \cdot \sigma_i^{\text{mean}} \end{aligned} \tag{26}$$

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

$$\tilde{x}_{ij}^{\text{sum}} = \frac{(x_{ij}^{\text{sum}} - \mu_i^{\text{sum}})}{\sigma_i^{\text{sum}}} = \frac{(d_i \cdot x_{ij}^{\text{mean}} - d_i \cdot \mu_i^{\text{mean}})}{d_i \cdot \sigma_i^{\text{mean}}} = \frac{(x_{ij}^{\text{mean}} - \mu_i^{\text{mean}})}{\sigma_i^{\text{mean}}} = \tilde{x}_{ij}^{\text{mean}}, \quad \forall i \in \mathcal{V}, \forall j = 1, \ldots, F, \tag{27}$$

is the same for both sum-aggregation and mean-aggregation.

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

Note that, BatchNorm does not have such an impact on degree information, since the normalization statistics are computed across all nodes (with different degrees) in each mini-batch per channel.