DATA-DRIVEN LEARNING OF GEOMETRIC SCATTERING NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph neural networks (GNNs) in general, and graph convolutional networks (GCN) in particular, often rely on low-pass graph filters to incorporate geometric information in the form of local smoothness over neighboring nodes. While this approach performs well on a surprising number of standard benchmarks, the efficacy of such models does not translate consistently to more complex domains, such as graph data in the biochemistry domain. We argue that these more complex domains require priors that encourage learning of band-pass and high-pass features rather than oversmoothed signals of standard GCN architectures. Here, we propose an alternative GNN architecture, based on a relaxation of recently proposed geometric scattering transforms, which consists of a cascade of graph wavelet filters. Our learned geometric scattering (LEGS) architecture adaptively tunes these wavelets and their scales to encourage band-pass features to emerge in learned representations. This results in a simplified GNN with significantly fewer learned parameters compared to competing methods. We demonstrate the predictive performance of our method on several biochemistry graph classification benchmarks, as well as the descriptive quality of its learned features in biochemical graph data exploration tasks. Our results show that the proposed LEGS network matches or outperforms popular GNNs, as well as the original geometric scattering construction, while also retaining certain mathematical properties of its handcrafted (nonlearned) design.

1 INTRODUCTION

Geometric deep learning has recently emerged as an increasingly prominent branch of machine learning in general, and deep learning in particular (Bronstein et al., 2017). It is based on the observation that many of the impressive achievements of neural networks come in applications where the data has an intrinsic geometric structure which can be used to inform network design and training procedures. For example, in computer vision, convolutional neural networks use the spatial organization of pixels to define convolutional filters that hierarchically aggregate local information at multiple scales that in turn encode shape and texture information in data and task-driven representations. Similarly, in time-series analysis, recurrent neural networks leverage memory mechanisms based on the temporal organization of input data to collect multiresolution information from local subsequences, which can be interpreted geometrically via tools from dynamical systems and spectral analysis. While these examples only leverage Euclidean spatiotemporal structure in data, they exemplify the potential benefits of incorporating information about intrinsic data geometry in neural network design and processing. Indeed, recent advances have further generalized the utilization of geometric information in neural networks design to consider non-Euclidean structures, with particular interest in graphs that represent data geometry, either directly given as input or constructed as an approximation of a data manifold.

At the core of geometric deep learning is the use of graph neural networks (GNNs) in general, and graph convolutional networks (GCNs) in particular, which ensure neuron activations follow the geometric organization of input data by propagating information across graph neighborhoods (Kipf & Welling, 2016; Veličković et al., 2018; Abu-El-Haija et al., 2019). However, recent work has shown the difficulty in generalizing these methods to more complex structures, identifying common problems and phrasing them in terms of oversmoothing (Li et al., 2018), oversquashing (Alon & Yahav, 2020) or under-reaching (Barceló et al., 2019). Using graph signal processing terminology

from Kipf & Welling (2016), these issues can be partly attributed to the limited construction of convolutional filters in GCNs and other common GNNs. Inspired by the filters learned in convolutional neural networks, GCNs consider node features as graph signals and aim to aggregate together information from neighboring nodes. However, unlike traditional convolutional networks, common aggregation methods in GCNs and other GNNs effectively perform local averaging or smoothing, which corresponds to low-pass filtering (i.e., retaining only low frequencies) in the graph spectral domain. This property extends to most common message passing and convolutional GNNs, which all smooth signals over rapidly increasingly large graph patches as the number of stacked layers grows. Known as oversmoothing, this prevents deeper cascades of such layers and hence, the representation of long-range dependencies within the graph.

Recently, an alternative approach was presented to provide deep geometric representation learning by generalizing Mallat's scattering transform (Mallat, 2012), originally proposed to provide a mathematical framework for understanding convolutional neural networks, to graphs (Gao et al., 2019; Gama et al., 2019a; Zou & Lerman, 2019) and manifolds Perlmutter et al. (2018). Similar to traditional scattering, which can be seen as a convolutional network with nonlearned wavelet filters, geometric scattering is defined as a GNN with handcrafted graph filters, typically constructed as diffusion wavelets over the input graph (Coifman & Maggioni, 2006), which are then cascaded with pointwise absolute-value nonlinearities. This wavelet cascade results in permutation equivariant node features that are typically aggregated via statistical moments over the graph nodes, as explained in detail in Sec. 2, to provide a permutation invariant graph-level representation. The efficacy of geometric scattering features in graph processing tasks was demonstrated in Gao et al. (2019), with both supervised learning and data exploration applications. Moreover, their handcrafted design enables rigorous study of their properties, such as stability to deformations and perturbations, and provides a clear understanding of the information extracted by them, which by design (e.g., the cascaded band-pass filters) goes beyond low frequencies to consider richer notions of regularity (Gama et al., 2019b; Perlmutter et al., 2019).

However, while graph scattering transforms provide effective universal feature extractors, their rigid handcrafted design does not allow for the automatic task-driven representation learning that naturally arises in traditional GNNs. To address this deficiency, recent work (Min et al., 2020) has proposed a hybrid scattering-GCN model for obtaining node-level representations, showing that integrating channels from both architectures alleviates the well-known oversmoothing problem and outperforms popular GNNs on node classification tasks. Here, we further study the incorporation of learned elements into geometric scattering, but focus on whole-graph representations with an emphasis on biochemical molecular graphs, where relatively large diameters and non-planar structures usually limit the effectiveness of traditional GNNs. Instead of the hybrid approach of Min et al. (2020), we propose a native neural network architecture for learned geometric scattering (LEGS), that directly modifies the scattering architecture from Gao et al. (2019); Perlmutter et al. (2019), via relaxations described in Sec. 3, to allow a task-driven adaptation of its wavelet configuration via backpropagation implemented in Sec. 4.

The resulting LEGS network balances the mathematical properties inherited from the scattering transform (as shown in Sec. 3) with the flexibility enabled by adaptive representation learning. The benefits of our construction over standard GNNs, as well as pure geometric scattering, are discussed and demonstrated on graph classification and regression tasks in Sec. 5. In particular, we find that our network maintains the robustness to small training sets present in graph scattering while improving classification on biological graph classification tasks, and we show that in tasks where the graphs have a large diameter relative to their size, learnable scattering features improve performance over competing methods.

2 Preliminaries: Geometric Scattering Features

Let $\mathcal{G} = (V, E, w)$ be a weighted graph with $V \coloneqq \{v_1, \ldots, v_n\}$ the set of nodes, $E \subset \{\{v_i, v_j\} \in V \times V, i \neq j\}$ the set of (undirected) edges and $w : E \to (0, \infty)$ assigning (positive) edge weights to the graph edges. Note that w can equivalently be considered as a function of $V \times V$, where we set the weights of non-adjacent node pairs to zero. We define a *graph signal* as a function $x : V \to \mathbb{R}$ on the nodes of $\mathcal{G}$ and aggregate them in a signal vector $\boldsymbol{x} \in \mathbb{R}^n$ with the ith entry being $x[v_i]$.

We define the *weighted adjacency matrix* $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ of the graph $\mathcal{G}$ as

$$W[v_i, v_j] \coloneqq \begin{cases} w(v_i, v_j) & \text{if } \{v_i, v_j\} \in E \\ 0 & \text{otherwise} \end{cases},$$

and the *degree matrix* $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ of $\mathcal{G}$ as $\boldsymbol{D} \coloneqq \operatorname{diag}(d_1, \dots, d_n)$ with $d_i \coloneqq \deg(v_i) \coloneqq \sum_{j=1}^{n} W[v_i, v_j]$ being the *degree* of the node v_i.

The geometric scattering transform (Gao et al., 2019) relies on a cascade of graph filters constructed from a row stochastic diffusion matrix $\boldsymbol{P} \coloneqq \frac{1}{2}\big(\boldsymbol{I}_n + \boldsymbol{W}\boldsymbol{D}^{-1}\big)$, which corresponds to transition probabilities of a lazy random walk Markov process. The laziness of the process signifies that at each step it has equal probability of either staying at the current node or transitioning to a neighbor, where transition probabilities in the latter case are determined by (normalized) edge weights. Scattering filters are then defined via the graph-wavelet matrices $\boldsymbol{\Psi}_j \in \mathbb{R}^{n \times n}$ of scale $j \in \mathbb{N}_0$, as

$$\begin{aligned} \boldsymbol{\Psi}_0 &\coloneqq \boldsymbol{I}_n - \boldsymbol{P}, \\ \boldsymbol{\Psi}_j &\coloneqq \boldsymbol{P}^{2^{j-1}} - \boldsymbol{P}^{2^j} = \boldsymbol{P}^{2^{j-1}}\big(\boldsymbol{I}_n - \boldsymbol{P}^{2^{j-1}}\big), \quad j \geq 1. \end{aligned} \tag{1}$$

These diffusion wavelet operators partition the frequency spectrum into dyadic frequency bands, which are then organized into a full wavelet filter bank $\mathcal{W}_J \coloneqq \{\boldsymbol{\Psi}_j, \boldsymbol{\Phi}_J\}_{0 \leq j \leq J}$, where $\boldsymbol{\Phi}_J \coloneqq \boldsymbol{P}^{2^J}$ is a pure low-pass filter, similar to the one used in GCNs. It is easy to verify that the resulting wavelet transform is invertible, since a simple sum of filter matrices in $\mathcal{W}_J$ yields the identity. Moreover, as discussed in Perlmutter et al. (2019), this filter bank forms a nonexpansive frame, which provides energy preservation guarantees as well as stability to perturbations, and can be generalized to a wider family of constructions that encompasses the variations of scattering transforms on graphs from Gama et al. (2019a;b) and Zou & Lerman (2019).

Given the wavelet filter bank $\mathcal{W}_J$, node-level scattering features are computed by stacking cascades of bandpass filters and element-wise absolute value nonlinearities to form

$$\boldsymbol{U}_p \boldsymbol{x} \coloneqq \boldsymbol{\Psi}_{j_m} |\boldsymbol{\Psi}_{j_{m-1}} \dots |\boldsymbol{\Psi}_{j_2} |\boldsymbol{\Psi}_{j_1} \boldsymbol{x}|| \dots |, \tag{2}$$

indexed (or parametrized) by the scattering path $p \coloneqq (j_1, \dots, j_m) \in \cup_{m \in \mathbb{N}} \mathbb{N}_0^m$ that determines the filter scales captured by each scattering coefficient. Then, a whole-graph scattering representation is obtained by aggregating together node-level features via statistical moments over the nodes of the graph (Gao et al., 2019). This construction yields the geometric scattering features

$$\boldsymbol{S}_{p,q} \boldsymbol{x} \coloneqq \sum_{i=1}^{n} |\boldsymbol{U}_p \boldsymbol{x}[v_i]|^q. \tag{3}$$

indexed by the scattering path p and moment order q. Finally, we note that it can be shown that the graph-level scattering transform $\boldsymbol{S}_{p,q}$ guarantees node-permutation invariance, while $\boldsymbol{U}_p$ is permutation equivariant (Perlmutter et al., 2019; Gao et al., 2019).

3 Relaxed geometric scattering construction to allow training

The geometric scattering construction, described in Sec. 2, can be seen as a particular GNN with handcrafted layers, rather than learned ones. This provides a solid mathematical framework for understanding the encoding of geometric information in GNNs, as shown in Perlmutter et al. (2019), while also providing effective unsupervised graph representation learning for data exploration, which also has some advantages even in supervised learning task, as shown in Gao et al. (2019). While the handcrafted design in Perlmutter et al. (2019); Gao et al. (2019) is not a priori amenable to task-driven tuning provided by end-to-end GNN training, we note that the cascade in Eq. 3 does conform to a neural network architecture suitable for backpropagation. Therefore, in this work, we propose to relax certain aspects of the scattering construction to enable training of a learned geometric scattering GNN, which preserves some of the mathematical properties established in Perlmutter et al. (2019) while also showing benefits in practice compared to purely handcrafted design and to fully learned one.

The first relaxation we provide here is on the construction of the diffusion matrix $\boldsymbol{P}$ that forms the lowpass filter used in the scattering construction to encode adaptive laziness by setting $\boldsymbol{P}_\alpha \coloneqq$

$\alpha \boldsymbol{I}_n + (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}$, where $\alpha \in [1/2, 1]$ controls the reluctance of the random walk to transition from one node to another. By enabling training of the laziness parameter, the learned transform will be able to control the locality and rate of information propagation in the filters constructed from this random walk. At this point, we note that one noticeable difference between the diffusion lowpass filter here and the one typically used in GCN and its variation is the symmetrization applied in Kipf & Welling (2016). However, Perlmutter et al. (2019) established that for the original construction, this is only a technical difference since $\boldsymbol{P}$ can be regarded as self-adjoint under an appropriate measure which encodes degree variations in the graph. This is then used to generate a Hilbert space $L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$ of graph signals with inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{D}^{-1/2}} := \langle \boldsymbol{D}^{-1/2}\boldsymbol{x}, \boldsymbol{D}^{-1/2}\boldsymbol{y} \rangle$. The following lemma shows that a similar property is retained for our adaptive lowpass filter $\boldsymbol{P}_\alpha$.

Lemma 1. *The matrix $\boldsymbol{P}_\alpha$ is self-adjoint on the Hilbert space $L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$ from Perlmutter et al. (2019).*

As a second relaxation, we propose to replace the handcrafted dyadic scales in Eq. 1 with an adaptive monotonic sequence of integer diffusion time scales $0 < t_1 < \cdots < t_J$, which can be selected or tuned via training. Then, an adaptive filter bank is constructed as $\mathcal{W}'_J := \{\boldsymbol{\Psi}'_j, \boldsymbol{\Phi}'_J\}_{j=0}^{J-1}$, with

$$\begin{aligned} \boldsymbol{\Phi}'_J &:= \boldsymbol{P}_\alpha^{t_J}, \\ \boldsymbol{\Psi}'_0 &:= \boldsymbol{I}_n - \boldsymbol{P}_\alpha^{t_1}, \\ \boldsymbol{\Psi}'_j &:= \boldsymbol{P}_\alpha^{t_j} - \boldsymbol{P}_\alpha^{t_{j+1}}, \quad 1 \leq j \leq J-1. \end{aligned} \tag{4}$$

The following theorem shows that for any selection of scales, the relaxed construction of $\mathcal{W}'_J$ constructs a nonexpansive frame, similar to the result from Perlmutter et al. (2019) shown for the original handcrafted construction.

Theorem 1. *There exist a constant $C > 0$ that only depends on t_1 and t_J such that for all $\boldsymbol{x} \in L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$,*

$$C\|\boldsymbol{x}\|^2_{\boldsymbol{D}^{-1/2}} \leqslant \|\boldsymbol{\Phi}'_J\boldsymbol{x}\|^2_{\boldsymbol{D}^{-1/2}} + \sum_{j=0}^{J} \|\boldsymbol{\Psi}'_j\boldsymbol{x}\|^2_{\boldsymbol{D}^{-1/2}} \leqslant \|\boldsymbol{x}\|^2_{\boldsymbol{D}^{-1/2}},$$

where the norm considered here is the one induced by the space $L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$.

Intuitively, the upper (i.e., nonexpansive) frame bound implies stability in the sense that small perturbations in the input graph signal will only result in small perturbations in the representation extracted by the constructed filter bank. Further, the lower frame bound ensures certain energy preservation by the constructed filter bank, thus indicating the nonexpansiveness is not implemented in a trivial fashion (e.g., by constant features independent of input signal).

In the next section we leverage the two relaxations described here to design a neural network architecture for learning the configuration $\alpha, t_1, \ldots, t_J$ of this relaxed construction via backpropagation through the resulting scattering filter cascade. The following theorem establishes that for any such configuration, extracted from $\mathcal{W}'_J$ via Eqs. 2-3, is permutation equivariant at the node-level and permutation invariant at the graph level. This guarantees that the extracted (in this case learned) features indeed encode intrinsic graph geometry rather than a priori indexation.

Theorem 2. *Let $\boldsymbol{U}'_p$ and $\boldsymbol{S}'_{p,q}$ be defined as in Eq. 2 and 3 (correspondingly), with the filters from $\mathcal{W}'_J$ with an arbitrary configuration $0 < \alpha < 1$, $0 < t_1 < \cdots < t_J$. Then, for any permutation Π over the nodes of $\mathcal{G}$, and any graph signal $\boldsymbol{x} \in L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$*

$$\boldsymbol{U}'_p \Pi \boldsymbol{x} = \Pi \boldsymbol{U}'_p \boldsymbol{x} \quad \textit{and} \quad \boldsymbol{S}'_{p,q}\Pi\boldsymbol{x} = \boldsymbol{S}'_{p,q}\boldsymbol{x} \qquad p \in \cup_{m\in\mathbb{N}}\mathbb{N}_0^m, q \in \mathbb{N}$$

where geometric scattering implicitly considers here the node ordering supporting its input signal.

We note that the results in Lemma 1 and Theorems 1-2, as well as their proofs, closely follow the theoretical framework proposed by Perlmutter et al. (2019). We carefully account here for the relaxed learned configuration, which replaces the originally handcrafted configuration there. For completeness, the adjusted proofs appear in Sec. A of the Appendix.

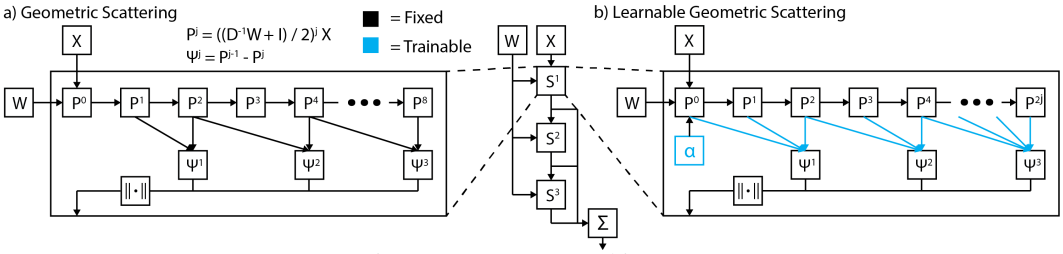

Figure 1: LEGSNet architecture

4 Learnable Geometric Scattering Network Architecture

In order to implement the relaxed geometric scattering construction (Sec. 3) via a trainable neural network, throughout this section, we consider an input graph signal $\boldsymbol{x} \in \mathbb{R}^n$ or, equivalently, a collection of graph signals $\boldsymbol{X} \in \mathbb{R}^{n \times N_{\ell-1}}$. The propagation of these signals can be divided into three major modules. First, a diffusion module implements the Markov process that forms the basis of the filter bank and transform, while allowing learning of the laziness parameter α. Then, a scattering module implements the filters and the corresponding cascade, while allowing the learning of the scales $t_1, \ldots, t_J$. Finally, the aggregation module collects the extracted features to provide a graph and produces the task-dependent output.

Building a diffusion process. We build a set of $m \in \mathbb{N}$ subsequent diffusion steps of the signal $\boldsymbol{x}$ by iteratively multiplying the diffusion matrix $\boldsymbol{P}_\alpha$ to the left of the signal, resulting in

$$\left[\boldsymbol{P}_\alpha \boldsymbol{x}, \boldsymbol{P}_\alpha^2 \boldsymbol{x}, \boldsymbol{P}_\alpha^3 \boldsymbol{x}, \ldots, \boldsymbol{P}_\alpha^m \boldsymbol{x}\right],$$

Since $\boldsymbol{P}_\alpha$ is often sparse, for efficiency reasons these filter responses are implemented via an RNN structure consisting of m RNN modules. Each module propagates the incoming hidden state $\boldsymbol{h}_{t-1}, t = 1, \ldots, m$ with $\boldsymbol{P}_\alpha$ with the readout $\boldsymbol{o}_t$ equal to the produced hidden state,

$$\boldsymbol{h}_t \coloneqq \boldsymbol{P}_\alpha \boldsymbol{h}_{t-1}, \quad \boldsymbol{o}_t \coloneqq \boldsymbol{h}_t.$$

Our architecture and theory enable the implementation of either trainable or nontrainable α, which we believe will be useful for future work as indicated, for example, in Gao & Ji (2019). However, in the applications considered here (see Sec. 5), we find that training α made training unstable and did not improve performance. Therefore, for simplicity, we leave it fixed as $\alpha = 1/2$ for the remainder of this work. In this case, the RNN portion of the network contains no trainable parameters, thus speeding up the computation, but still enables a convenient gradient flow back to the model input.

Learning diffusion filter bank. Next, we consider the selection of $J \leq m$ diffusion scales for the relaxed filter bank construction with the wavelets defined according to Eq. 5. We found this was the most influential part of the architecture. We experimented with methods of increasing flexibility:

1. Selection of $\{t_j\}_{j=1}^{J-1}$ as dyadic scales (as in Sec. 2 and Eq. 1), fixed for all datasets (LEGS-FIXED),
2. Selection of each t_j using softmax and sorting by j, learnable per model (LEGS-FCN and LEGS-RBF, depending on output layer explained below).

For the softmax selection, we use a selection matrix $\boldsymbol{F} \in \mathbb{R}^{J \times m}$, where each row $\boldsymbol{F}_{(j,\cdot)}, j = 1, \ldots, J$ is dedicated to identifying the diffusion scale of the wavelet $\boldsymbol{P}_\alpha^{t_j}$ via a one-hot encoding. This is achieved by setting

$$\boldsymbol{F} \coloneqq \mathrm{softmax}(\boldsymbol{\Theta}) = [\mathrm{softmax}(\boldsymbol{\theta}_1), \mathrm{softmax}(\boldsymbol{\theta}_2), \ldots, \mathrm{softmax}(\boldsymbol{\theta}_J)]^T$$

where $\boldsymbol{\theta}_j \in \mathbb{R}^m$ constitute the rows of the trainable weight matrix $\boldsymbol{\Theta}$. While this construction may not strictly guarantee an exact one-hot encoding, we assume that the softmax activations yield a sufficient approximation. Further, without loss of generality, we assume that the rows of $\boldsymbol{F}$ are ordered according to the position of the leading "one" activated in every row. In practice, this can

be easily enforced by reordering the rows. We now construct the filter bank $\widetilde{\mathcal{W}}_{\boldsymbol{F}} := \{\widetilde{\boldsymbol{\Psi}}_j, \widetilde{\boldsymbol{\Phi}}_J\}_{j=0}^{J-1}$ with the filters

$$\begin{aligned}
\widetilde{\boldsymbol{\Phi}}_J \boldsymbol{x} &= \sum\nolimits_{t=1}^{m} \boldsymbol{F}_{(J,t)} \boldsymbol{P}_\alpha^t \boldsymbol{x}, \\
\widetilde{\boldsymbol{\Psi}}_0 \boldsymbol{x} &= \boldsymbol{I}_n - \sum\nolimits_{t=1}^{m} \boldsymbol{F}_{(1,t)} \boldsymbol{P}_\alpha^t \boldsymbol{x} \\
\widetilde{\boldsymbol{\Psi}}_j \boldsymbol{x} &= \sum\nolimits_{t=1}^{m} \left[\boldsymbol{F}_{(j,t)} \boldsymbol{P}_\alpha^t \boldsymbol{x} - \boldsymbol{F}_{j+1,t} \boldsymbol{P}_\alpha^t \boldsymbol{x}\right] \qquad 1 \leq j \leq J-1
\end{aligned} \tag{5}$$

matching and implementing the construction of $\mathcal{W}'_J$ from Eq. 4.

Aggregating and classifying scattering features. While multiple approaches may be applied to aggregate node-level features into graph-level features, here we follow the statistical-moment aggregation explained in Secs. 2-3. As shown in Gao et al. (2019) on graph classification, this aggregation works particularly well in conjunction with support vector machines (SVMs) based on the radial basis function (RBF) kernel. Here, we consider two configurations for the task-dependent output layer of the network, either using a small neural network with two fully connected layers, which we denote LEGS-FCN, or using a modified RBF network (Broomhead & Lowe, 1988), which we denote LEGS-RBF, to produce the final classification. The latter configuration more accurately processes scattering features as shown in Table 2. Our RBF network works by first initializing a fixed number of movable anchor points. Then, for every point, new features are calculated based on the radial distances to these anchor points. In previous work on radial basis networks these anchor points were initialized independent of the data. We found that this led to training issues if the range of the data was not similar to the initialization of the centers. Instead, we first use a batch normalization layer to constrain the scale of the features and then pick anchors randomly from the initial features of the first pass through our data. This gives an RBF-kernel network with anchors that are always in the range of the data. Our RBF layer is then $\text{RBF}(\boldsymbol{x}) = \phi(\|\text{BatchNorm}(\boldsymbol{x}) - \boldsymbol{c}\|)$ with $\quad \phi(\boldsymbol{x}) = e^{-\|\boldsymbol{x}\|^2}$.

5 Empirical Results

Here we show results of LEGSNet on whole graph classification and graph regression tasks, that arise in a variety of contexts, with emphasis on the more complex biochemic datasets. We use biochemical graph datasets as they represent a new challenge in the field of graph learning. Unlike other types of data, these datasets do not exhibit the small-world structure of social datasets and may have large graph diameters for their size. Further, the connectivity patterns of biomolecules are very irregular due to 3D folding and long range connections, and thus ordinary local node aggregation methods may miss such connectivity differences.

Table 1: Dataset statistics, diameter, nodes, edges, and clustering coefficient averaged over graphs.

	# Graphs	# Classes	Diameter	Nodes	Edges	Clust. Coeff
DD	1178	2	19.81	284.32	715.66	0.48
ENZYMES	600	6	10.92	32.63	62.14	0.45
MUTAG	188	2	8.22	17.93	19.79	0.00
NCI1	4110	2	13.33	29.87	32.30	0.00
NCI109	4127	2	13.14	29.68	32.13	0.00
PROTEINS	1113	2	11.62	39.06	72.82	0.51
PTC	344	2	7.52	14.29	14.69	0.01

5.1 Whole Graph Classification

We perform whole graph classification by using eccentricity and clustering coefficient as node features as is done in Gao et al. (2019). We compare against graph convolutional networks (GCN) (Kipf & Welling, 2016), GraphSAGE (Hamilton et al., 2017), fixed geometric scattering with a support vector machine classifier (GS-SVM) as in Gao et al. (2019), and a baseline which is a 2-layer neural network on the features averaged across nodes (disregarding graph structure). These comparisons are meant to inform when including learnable graph scattering features are helpful in extracting whole graph features. Specifically, we are interested in the types of graph datasets where existing graph neural network performance can be improved upon with scattering features. We evaluate these methods across 7 benchmark biochemical datasets DD, ENZYMES, MUTAG, NCI1, NCI109, PROTEINS, and PTC where the goal is to classify between two or more classes of compounds with hundreds to thousands of graphs and tens to hundreds of nodes (See Table 1). For completeness we also show results on six social network datasets in Table S2. For more specific information on individual datasets see Appendix B. We use 10-fold cross validation on all models which is elaborated on in Appendix C.

Table 2: Mean $\pm$ standard deviation test set accuracy on biochemical datasets.

	LEGS-RBF	LEGS-FCN	LEGS-FIXED	GCN	GraphSAGE	GS-SVM	Baseline
DD	72.58 ± 3.35	72.07 ± 2.37	69.09 ± 4.82	67.82 ± 3.81	66.37 ± 4.45	72.66 ± 4.94	$\mathbf{75.98 \pm 2.81}$
ENZYMES	36.33 ± 4.50	$\mathbf{38.50 \pm 8.18}$	32.33 ± 5.04	31.33 ± 6.89	15.83 ± 9.10	27.33 ± 5.10	20.50 ± 5.99
MUTAG	33.51 ± 4.34	82.98 ± 9.85	81.84 ± 11.24	79.30 ± 9.66	81.43 ± 11.64	$\mathbf{85.09 \pm 7.44}$	79.80 ± 9.92
NCI1	$\mathbf{74.26 \pm 1.53}$	70.83 ± 2.65	71.24 ± 1.63	60.80 ± 4.26	57.54 ± 3.33	69.68 ± 2.38	56.69 ± 3.07
NCI109	$\mathbf{72.47 \pm 2.11}$	70.17 ± 1.46	69.25 ± 1.75	61.30 ± 2.99	55.15 ± 2.58	68.55 ± 2.06	57.38 ± 2.20
PROTEINS	70.89 ± 3.91	71.06 ± 3.17	67.30 ± 2.94	$\mathbf{74.03 \pm 3.20}$	71.87 ± 3.50	70.98 ± 2.67	73.22 ± 3.76
PTC	$\mathbf{57.26 \pm 5.54}$	56.92 ± 9.36	54.31 ± 6.92	56.34 ± 10.29	55.22 ± 9.13	56.96 ± 7.09	56.71 ± 5.54

LEGS outperforms on biological datasets. A somewhat less explored domain for GNNs is in biochemical graphs that represent molecules and tend to be overall smaller and less connected (see Tables 1 and S1) than social networks. In particular we find that LEGSNet outperforms other methods by a significant margin on biochemical datasets with relatively small but high diameter graphs (NCI1, NCI109, ENZYMES, PTC), as shown in Table 2, whereas on the social network datasets GCN and GraphSage perform quite well (see Table S2). On extremely small graphs we find that GS-SVM performs best, which is expected as other methods with more parameters can easily overfit the data. We reason that the performance increases exhibited by LEGSNet, and to a lesser extent GS-SVM, on these chemical and biological benchmarks is due the ability of geometric scattering to compute complex connectivity features via its multiscale diffusion wavelets. Thus, methods that rely on a scattering construction would in general perform better, with the flexibility and trainability LEGSNet giving it an edge on most tasks.

LEGS preserves enzyme exchange preferences while increasing performance One advantage of geometric scattering over other graph embedding techniques lies in the rich information present within the scattering feature space. This was demonstrated in Gao et al. (2019) by showing the embeddings created through graph scattering can be used to accurately infer intergraph relationships. Scattering features of enzyme graph within the ENZYMES dataset (Borgwardt et al., 2005) possessed sufficient global information to recreate the enzyme class exchange preferences, observed empirically by Cuesta et al. (2015), using only linear methods of analysis, and despite working with a much smaller and artificially balanced dataset. We demonstrate here that LEGSNet retains similar descriptive capabilities, as shown in Figure 2 via chord diagrams where each exchange preference between enzyme classes (estimated as suggested in Gao et al., 2019) is represented as ribbon of the corresponding size. Our results here (and in Table S5, which provides complementary quantitative comparison) show that, with relaxations on the scattering parameters, LEGS-FCN achieves better classification accuracy than both LEGS-FIXED and GCN (see Table 1) while also retaining a more descriptive embedding that maintains the global structure of relations between enzyme classes.

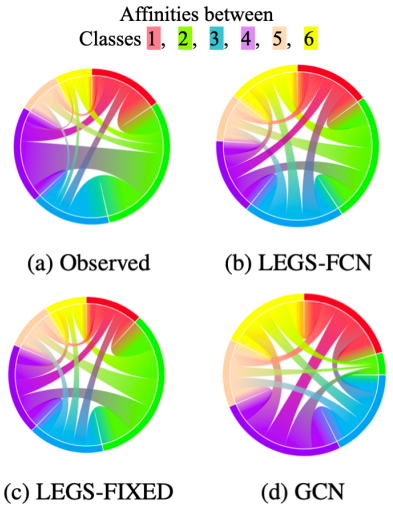

Figure 2: Enzyme class exchange preferences empirically observed in Cuesta et al. (2015), and estimated from LEGS and GCN embeddings.

We ran two varieties of LEGSNet on the ENZYMES dataset: LEGS-FIXED and LEGS-FCN, which allows the diffusion scales to be learned. For comparison, we also ran a standard GCN whose graph embeddings were obtained via mean pooling. To infer enzyme exchange preferences from their embeddings, we followed Gao et al. (2019) in defining the distance from an enzyme e to the enzyme class EC_j as $\text{dist}(e, \text{EC}_j) := \|v_e - \text{proj}_{C_j}(v_e)\|$, where v_i is the embedding of e, and C_j is the PCA subspace of the enzyme feature vectors within EC_j. The distance between the enzyme classes EC_i and EC_j is the average of the individual distances, $\text{mean}\{\text{dist}(e, \text{EC}_j) : e \in \text{EC}_i\}$. From here, the affinity between two enzyme classes is computed as $\text{pref}(\text{EC}_i, \text{EC}_j) = w_i / \min(\frac{D_{i,i}}{D_{i,j}}, \frac{D_{j,j}}{D_{j,i}})$, where w_i is the percentage of enzymes in class i which are closer to another class than their own, and $D_{i,j}$ is the distance between EC_i and EC_j.

Robustness to reduced training set size. We remark that similar to the robustness shown in (Gao et al., 2019) for handcrafted scattering, LEGSNet is able to maintain accuracy even when the training set size is shrunk to as low as 20% of the dataset, with a median decrease of 4.7% accuracy as when 80% of the data is used for training, as discussed in the supplement (see Table S3).

5.2 GRAPH REGRESSION

We next use a graph regression task from the critical assessment of structure prediction (CASP) challenge (Moult et al., 2018). The main task is to score protein structure prediction/simulation models in terms of the discrepancy between their predicted structure and the actual structure of the protein (which is known a priori). The accuracy of such 3D structure predictions are evaluated using a variety of metrics, but we focus on the global distance test (GDT) score (Modi et al., 2016). The GDT score measures the similarity between tertiary structures of two proteins with amino-acid correspondence. A higher score means two structures are more similar. For a set of predicted 3D structures for a protein, we would like to score their quality as quantified by the GDT score.

Table 3: Train and test set mean squared error on CASP GDT regression task over three seeds.

$(\mu \pm \sigma)$	Train MSE	Test MSE
LEGS-FCN	**134.34 ± 8.62**	**144.14 ± 15.48**
LEGS-RBF	140.46 ± 9.76	152.59 ± 14.56
LEGS-FIXED	136.84 ± 15.57	160.03 ± 1.81
GCN	289.33 ± 15.75	303.52 ± 18.90
GraphSAGE	221.14 ± 42.56	219.44 ± 34.84
Baseline	393.78 ± 4.02	402.21 ± 21.45

For this task we use the CASP12 dataset (Moult et al., 2018) and preprocess the data similarly to Ingraham et al. (2019), creating a KNN graph between proteins based on the 3D coordinates of each amino acid. From this KNN graph we regress against the GDT score. We evaluate on 12 proteins from the CASP12 dataset and choose random (but consistent) splits with 80% train, 10% validation, and 10% test data out of 4000 total structures. We are only concerned with structure similarity so use no non-structural node features.

LEGSNet outperforms on all CASP targets Across all CASP targets we find that LEGSNet significantly outperforms GNN and baseline methods (See Table S4). This performance improvement is particularly stark on the easiest structures (measured by average GDT) but is consistent across all structures. In Figure 3 we show the relationship between percent improvement of LEGSNet over the GCN model and the average GDT score across the target structures. We draw attention to target t0879, where LEGSNet shows the greatest improvement over other methods. This target has long range dependencies (Ovchinnikov et al., 2018) as it exhibits metal coupling (Li et al., 2015) creating long range connections over the sequence. Since other methods are unable to model these long range connections LEGSNet is particularly important on these more difficult to model targets.

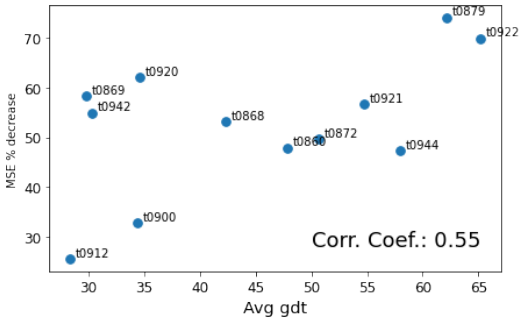

Figure 3: CASP dataset LEGS-FCN % improvement over GCN in MSE of GDT prediction vs. Average GDT score.

6 CONCLUSION

In this work we have established a relaxation from geometric scattering with strong guarantees to a progressively more flexible network with better performance, but fewer guarantees. Allowing the network to choose data-driven diffusion scales leads to improved performance particularly on biochemical datasets, while keeping strong guarantees on extracted features. This parameterization has advantages in representing long range connections with a small number of weights, which are necessary in complex biochemical data. This also opens the possibility to provide additional relaxation to enable node-specific tuning via attention mechanisms, which we regard as an exciting future direction, but out of scope for the current work.

REFERENCES

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

Pablo Barceló, Egor V Kostylev, Mikael Monet, Jorge Pérez, Juan Reutter, and Juan Pablo Silva. The logical expressiveness of graph neural networks. In *6th International Conference on Learning Representations*, 2019.

K. M. Borgwardt, C. S. Ong, S. Schonauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(Suppl 1):i47–i56, June 2005. ISSN 1367-4803, 1460-2059. doi: 10.1093/bioinformatics/bti1007.

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond Euclidean data. *IEEE Signal Process. Mag.*, 34(4):18–42, 2017. ISSN 1053-5888. doi: 10.1109/MSP.2017.2693418.

D. S. Broomhead and D Lowe. Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks. *R. Signals Raar Establ.*, Memorandum 4148, 1988.

Ronald R. Coifman and Mauro Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, July 2006. ISSN 10635203. doi: 10.1016/j.acha.2006.04.004.

Sergio Martinez Cuesta, Syed Asad Rahman, Nicholas Furnham, and Janet M. Thornton. The Classification and Evolution of Enzyme Function. *Biophysical Journal*, 109(6):1082–1086, September 2015. ISSN 00063495. doi: 10.1016/j.bpj.2015.04.020.

Paul D. Dobson and Andrew J. Doig. Distinguishing Enzyme Structures from Non-enzymes Without Alignments. *Journal of Molecular Biology*, 330(4):771–783, July 2003. ISSN 0022-2836. doi: 10.1016/S0022-2836(03)00628-4.

Fernando Gama, Joan Bruna, and Alejandro Ribeiro. Diffusion Scattering Transforms on Graphs. *Int. Conf. Mach. Learn.*, 2019a.

Fernando Gama, Joan Bruna, and Alejandro Ribeiro. Stability of Graph Scattering Transforms. *Adv. Neural Inf. Process. Syst. 32*, 2019b.

Feng Gao, Guy Wolf, and Matthew Hirn. Geometric scattering for graph data analysis. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2122–2131, 2019.

Hongyang Gao and Shuiwang Ji. Graph U-Nets. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. *Adv. Neural Inf. Process. Syst. 31*, 2017.

John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative Models for Graph-Based Protein Design. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 15820–15831, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations*, 2015.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *4th Int. Conf. Mach. Learn.*, September 2016.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pp. 3538–3545. Association for the Advancement of Artificial Intelligence, 2018.

Wenfei Li, Jun Wang, Jian Zhang, and Wei Wang. Molecular simulations of metal-coupled protein folding. *Current Opinion in Structural Biology*, 30:25–31, February 2015. ISSN 0959-440X. doi: 10.1016/j.sbi.2014.11.006.

Stéphane Mallat. Group Invariant Scattering. *Commun. Pure Appl. Math.*, LXV:1331–1398, 2012.

Yimeng Min, Frederik Wenkel, and Guy Wolf. Scattering gcn: Overcoming oversmoothness in graph convolutional networks. *arXiv preprint arXiv:2003.08414*, 2020.

Vivek Modi, Qifang Xu, Sam Adhikari, and Roland L. Dunbrack. Assessment of Template-Based Modeling of Protein Structure in CASP11. *Proteins*, 84(Suppl 1):200–220, September 2016. ISSN 0887-3585. doi: 10.1002/prot.25049.

John Moult, Krzysztof Fidelis, Andriy Kryshtafovych, Torsten Schwede, and Anna Tramontano. Critical assessment of methods of protein structure prediction (CASP)—Round XII. *Proteins Struct. Funct. Bioinforma.*, 86(S1):7–15, 2018. ISSN 1097-0134. doi: 10.1002/prot.25415.

Sergey Ovchinnikov, Hahnbeom Park, David E. Kim, Frank DiMaio, and David Baker. Protein structure prediction using Rosetta in CASP12. *Proteins Struct. Funct. Bioinforma.*, 86(S1):113–121, 2018. ISSN 1097-0134. doi: 10.1002/prot.25390.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pp. 8026–8037, 2019.

Michael Perlmutter, Guy Wolf, and Matthew Hirn. Geometric scattering on manifolds. In *NeurIPS 2018 Workshop on Integration of Deep Learning Theories*, 2018.

Michael Perlmutter, Feng Gao, Guy Wolf, and Matthew Hirn. Understanding graph neural networks with asymmetric geometric scattering transforms. *arXiv preprint arXiv:1911.06253*, 2019.

H. Toivonen, A. Srinivasan, R. D. King, S. Kramer, and C. Helma. Statistical evaluation of the Predictive Toxicology Challenge 2000-2001. *Bioinformatics*, 19(10):1183–1193, July 2003. ISSN 1367-4803, 1460-2059. doi: 10.1093/bioinformatics/btg130.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *6th Int. Conf. Learn. Represent.*, 2018.

Nikil Wale, Ian A Watson, and George Karypis. Comparison of Descriptor Spaces for Chemical Compound Retrieval and Classification. *Knowl. Inf. Syst.*, 2008.

Pinar Yanardag and S.V.N. Vishwanathan. Deep Graph Kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15*, pp. 1365–1374, Sydney, NSW, Australia, 2015. ACM Press. ISBN 978-1-4503-3664-2. doi: 10.1145/2783258.2783417.

Dongmian Zou and Gilad Lerman. Graph convolutional neural networks via scattering. *Applied and Computational Harmonic Analysis*, 2019. ISSN 10635203. doi: 10.1016/j.acha.2019.06.003.

APPENDIX

A PROOFS FOR SECTION 3

A.1 PROOF OF LEMMA 1

Let $\boldsymbol{M}_\alpha = \boldsymbol{D}^{-1/2}\boldsymbol{P}_\alpha\boldsymbol{D}^{1/2}$ then it can be verified that $\boldsymbol{M}_\alpha$ is a symmetric conjugate of $\boldsymbol{P}_\alpha$, and by construction is self-adjoint with respect to the standard inner product of $L^2(\mathcal{G})$. Let $\boldsymbol{x}, \boldsymbol{y} \in L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$ then we have

$$\begin{aligned}
\langle \boldsymbol{P}_\alpha \boldsymbol{x}, \boldsymbol{y}\rangle_{\boldsymbol{D}^{-1/2}} &= \langle \boldsymbol{D}^{-1/2}\boldsymbol{P}_\alpha\boldsymbol{x}, \boldsymbol{D}^{-1/2}\boldsymbol{y}\rangle \\
&= \langle \boldsymbol{D}^{-1/2}\boldsymbol{D}^{1/2}\boldsymbol{M}_\alpha\boldsymbol{D}^{-1/2}\boldsymbol{x}, \boldsymbol{D}^{-1/2}\boldsymbol{y}\rangle \\
&= \langle \boldsymbol{M}_\alpha\boldsymbol{D}^{-1/2}\boldsymbol{x}, \boldsymbol{D}^{-1/2}\boldsymbol{y}\rangle \\
&= \langle \boldsymbol{D}^{-1/2}\boldsymbol{x}, \boldsymbol{M}_\alpha\boldsymbol{D}^{-1/2}\boldsymbol{y}\rangle \\
&= \langle \boldsymbol{D}^{-1/2}\boldsymbol{x}, \boldsymbol{D}^{-1/2}\boldsymbol{D}^{1/2}\boldsymbol{M}_\alpha\boldsymbol{D}^{-1/2}\boldsymbol{y}\rangle \\
&= \langle \boldsymbol{D}^{-1/2}\boldsymbol{x}, \boldsymbol{D}^{-1/2}\boldsymbol{P}_\alpha\boldsymbol{y}\rangle \\
&= \langle \boldsymbol{x}, \boldsymbol{P}_\alpha\boldsymbol{y}\rangle_{\boldsymbol{D}^{-1/2}},
\end{aligned}$$

which gives the result of the lemma. □

A.2 PROOF OF THEOREM 1

As shown in the previous proof (Sec. A.1), $\boldsymbol{P}_\alpha$ has a symmetric conjugate $\boldsymbol{M}_\alpha$. Given the eigendecomposition $\boldsymbol{M}_\alpha = Q\Lambda Q^T$, we can write $\boldsymbol{P}_\alpha^t = \boldsymbol{D}^{1/2}Q\Lambda^t Q^T\boldsymbol{D}^{-1/2}$, giving the eigendecomposition of the propagated diffusion matrices. Furthermore, it can be verified that the eigenvalues on the diagonal of Λ are nonnegative. Briefly, this results from graph Laplacian eigenvalues being within the range $[0, 1]$, which means those of $\boldsymbol{W}\boldsymbol{D}^{-1}$ are in $[-1, 1]$, which combined with $1/2 \leq \alpha \leq 1$ result in $\lambda_i \coloneqq [\Lambda]_{ii} \in [0, 1]$ for every j. Next, given this decomposition we can write:

$$\begin{aligned}
\boldsymbol{\Phi}'_J &= \boldsymbol{D}^{1/2}Q\Lambda^{t_J}Q^T\boldsymbol{D}^{-1/2}, \\
\boldsymbol{\Psi}'_j &= \boldsymbol{D}^{1/2}Q(\Lambda^{t_j} - \Lambda^{t_{j+1}})Q^T\boldsymbol{D}^{-1/2}, \quad 0 \leq j \leq J-1.
\end{aligned}$$

where we set $t_0 = 0$ to simplify notations. Then, we have:

$$\begin{aligned}
\|\boldsymbol{\Phi}'_J\boldsymbol{x}\|^2_{D^{-1/2}} &= \langle \boldsymbol{\Phi}'_J\boldsymbol{x}, \boldsymbol{\Phi}'_J\boldsymbol{x}\rangle_{D^{-1/2}} \\
&= \langle D^{-1/2}\boldsymbol{D}^{1/2}Q\Lambda^{t_J}Q^T\boldsymbol{D}^{-1/2}\boldsymbol{x}, D^{-1/2}\boldsymbol{D}^{1/2}Q\Lambda^{t_J}Q^T\boldsymbol{D}^{-1/2}\boldsymbol{x}\rangle \\
&= \boldsymbol{x}^T\boldsymbol{D}^{-1/2}Q\Lambda^{t_J}Q^TQ\Lambda^{t_J}Q^T\boldsymbol{D}^{-1/2}\boldsymbol{x} = (\boldsymbol{x}^T\boldsymbol{D}^{-1/2}Q\Lambda^{t_J})(\Lambda^{t_J}Q^T\boldsymbol{D}^{-1/2}\boldsymbol{x}) \\
&= \|\Lambda^{t_J}Q^T\boldsymbol{D}^{-1/2}\boldsymbol{x}\|_2^2
\end{aligned}$$

Further, since Q is orthogonal (as it is constructed from an eigenbasis of a symmetric matrix), if we consider a change of variable to $\boldsymbol{y} = Q^T\boldsymbol{D}^{-1/2}\boldsymbol{x}$, we have $\|\boldsymbol{x}\|^2_{\boldsymbol{D}^{-1/2}} = \|\boldsymbol{D}^{-1/2}\boldsymbol{x}\|_2^2 = \|\boldsymbol{y}\|_2^2$ while $\|\boldsymbol{\Phi}'_J\boldsymbol{x}\|^2_{D^{-1/2}} = \|\Lambda^{t_J}\boldsymbol{y}\|_2^2$. Similarly, we can also reformulate the operation of other filters in terms of diagonal matrices applied to $\boldsymbol{y}$ as $\mathcal{W}'_J$ as $\|\boldsymbol{\Psi}'_j\boldsymbol{x}\|^2_{D^{-1/2}} = \|(\Lambda^{t_j} - \Lambda^{t_{j+1}})\boldsymbol{y}\|_2^2$.

Given the reformulation in terms of $\boldsymbol{y}$ and standard $L^2(\mathcal{G})$, we can now write

$$\|\Lambda^{t_J}\boldsymbol{y}\|_2^2 + \sum_{j=0}^{J-1}\|(\Lambda^{t_j} - \Lambda^{t_{j+1}})\boldsymbol{y}\|_2^2 = \sum_{i=1}^{n}\boldsymbol{y}_i^2 \cdot \left(\lambda^{2t_J} + \sum\nolimits_{j=0}^{J-1}(\lambda_i^{t_j} - \lambda_i^{t_{j+1}})^2\right).$$

Then, since $0 \leq \lambda_i \leq 1$ and $0 = t_0 < t_1 < \cdots < t_J$ we have

$$\lambda^{2t_J} + \sum_{j=0}^{J-1}(\lambda_i^{t_j} - \lambda_i^{t_{j+1}})^2 \leq \left(\lambda^{t_J} + \sum_{j=0}^{J-1}\lambda_i^{t_j} - \lambda_i^{t_{j+1}}\right)^2 = \left(\lambda^{t_J} + \lambda_i^{t_0} - \lambda_i^{t_J}\right)^2 = 1,$$

which yields the upper bound $\|\Lambda^{t_J}\boldsymbol{y}\|_2^2 + \sum_{j=0}^{J-1}\|(\Lambda^{t_j} - \Lambda^{t_{j+1}})\boldsymbol{y}\|_2^2 \leq \|\boldsymbol{y}\|_2^2$. On the other hand, since $t_1 > 0 = t_0$, then we also have

$$\lambda^{2t_J} + \sum_{j=0}^{J-1}(\lambda_i^{t_j} - \lambda_i^{t_{j+1}})^2 \geq \lambda^{2t_J} + (1 - \lambda_i^{t_1})^2$$

and therefore, by setting $C := \min_{0\leq\xi\leq 1}(\xi^{2t_J} + (1-\xi^{t_1})^2) > 0$, whose positivity is not difficult to verify, we get the lower bound $\|\Lambda^{t_J}\boldsymbol{y}\|_2^2 + \sum_{j=0}^{J-1}\|(\Lambda^{t_j} - \Lambda^{t_{j+1}})\boldsymbol{y}\|_2^2 \geq C\|\boldsymbol{y}\|_2^2$. Finally, applying applying the reverse change of variable to $\boldsymbol{x}$ and $L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$ yields the result of the theorem. □

A.3 Proof of Theorem 2

Denote the permutation group on n elements as S_n, then for a permutation $\Pi \in S_n$ we let $\overline{\mathcal{G}} = \Pi(\mathcal{G})$ be the graph obtained by permuting the vertices of $\mathcal{G}$ with Π. The corresponding permutation operation on a graph signal $\boldsymbol{x} \in L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$ gives a signal $\Pi\boldsymbol{x} \in L^2(\overline{\mathcal{G}}, \boldsymbol{D}^{-1/2})$, which we implicitly considered in the statement of the theorem, without specifying these notations for simplicity. Rewriting the statement of the theorem more rigorously with the introduced notations, we aim to show that $\overline{\boldsymbol{U}}_p'\Pi\boldsymbol{x} = \Pi\boldsymbol{U}_p'\boldsymbol{x}$ and $\overline{\boldsymbol{S}}_{p,q}'\Pi\boldsymbol{x} = \boldsymbol{S}_{p,q}'\boldsymbol{x}$ under suitable conditions, where the operation $\boldsymbol{U}_p'$ from $\mathcal{G}$ on the permuted graph $\overline{\mathcal{G}}$ is denoted here by $\overline{\boldsymbol{U}}_p'$ and likewise for $\boldsymbol{S}_{p,q}'$ we have $\overline{\boldsymbol{S}}_{p,q}'$.

We start by showing $\boldsymbol{U}_p'$ is permutation equivariant. First, we notice that for any Ψ_j, $0 < j < J$ we have that $\overline{\Psi}_j\Pi\boldsymbol{x} = \Pi\Psi_j\boldsymbol{x}$, as for $1 \leq j \leq J-1$

$$\begin{aligned}\overline{\Psi}_j\Pi\boldsymbol{x} &= (\Pi\boldsymbol{P}^{t_j}\Pi^T - \Pi\boldsymbol{P}^{t_{j+1}}\Pi^T)\Pi\boldsymbol{x}\\ &= \Pi(\boldsymbol{P}^{t_j} - \boldsymbol{P}^{t_{j+1}})\boldsymbol{x}\\ &= \Pi\Psi_j\boldsymbol{x}.\end{aligned}$$

Similar reasoning also holds for $j \in \{0, J\}$. Further, notice that for the element-wise nature of the absolute value nonlinearity yields $|\Pi\boldsymbol{x}| = \Pi|\boldsymbol{x}|$ for any permutation matrix Π. Using these two observations, it follows inductively that

$$\begin{aligned}\overline{\boldsymbol{U}}_p'\Pi\boldsymbol{x} :=& \boldsymbol{\Psi}_{j_m}'|\boldsymbol{\Psi}_{j_{m-1}}'\ldots|\boldsymbol{\Psi}_{j_2}'|\boldsymbol{\Psi}_{j_1}'\Pi\boldsymbol{x}||\ldots|\\ =& \boldsymbol{\Psi}_{j_m}'|\boldsymbol{\Psi}_{j_{m-1}}'\ldots|\boldsymbol{\Psi}_{j_2}'\Pi|\boldsymbol{\Psi}_{j_1}'\boldsymbol{x}||\ldots|\\ &\vdots\\ =& \Pi\boldsymbol{\Psi}_{j_m}'|\boldsymbol{\Psi}_{j_{m-1}}'\ldots|\boldsymbol{\Psi}_{j_2}'|\boldsymbol{\Psi}_{j_1}'\boldsymbol{x}||\ldots|\\ =& \Pi\boldsymbol{U}_p'\boldsymbol{x}.\end{aligned}$$

To show $\boldsymbol{S}_{p,q}'$ is permutation invariant, first notice that for any statistical moment $q > 0$, we have $|\Pi\boldsymbol{x}|^q = \Pi|\boldsymbol{x}|^q$ and further as sums are commutative, $\sum_j(\Pi\boldsymbol{x})_j = \sum_j \boldsymbol{x}_j$. We then have

$$\overline{\boldsymbol{S}}_{p,q}'\Pi\boldsymbol{x} = \sum_{i=1}^n|\overline{\boldsymbol{U}}_p'\Pi\boldsymbol{x}[v_i]|^q = \sum_{i=1}^n|\Pi\boldsymbol{U}_p'\boldsymbol{x}[v_i]|^q = \sum_{i=1}^n|\boldsymbol{U}_p'\boldsymbol{x}[v_i]|^q = \boldsymbol{S}_{p,q}'\boldsymbol{x},$$

which, together with the previous result, completes the proof of the theorem. □

B Datasets

In this section we further analyze individual datasets. Relating composition of the dataset as shown in Table S1 to the relative performance of our models as shown in Table S2.

DD Dobson & Doig (2003): Is a dataset extracted from the protein data bank (PDB) of 1178 high resolution proteins. The task is to distinguish between enzymes and non-enzymes. Since these are high resolution structures, these graphs are significantly larger than those found in our other biochemical datasets with a mean graph size of 284 nodes with the next largest biochemical dataset with a mean size of 39 nodes.

ENZYMES Borgwardt et al. (2005): Is a dataset of 600 enzymes divided into 6 balanced classes of 100 enzymes each. As we analyzed in the main text, scattering features are better able to preserve the structure between classes. LEGS-FCN slightly relaxes this structure but improves accuracy from 32 to 39% over LEGS-FIXED.

NCI1, NCI109 Wale et al. (2008): Contains slight variants of 4100 chemical compounds encoded as graphs. Each compound is separated into one of two classes based on its activity against non-small cell lung cancer and ovarian cancer cell lines. Graphs in this dataset are 30 nodes with a similar number of edges. This makes for long graphs with high diameter.

PROTEINS Borgwardt et al. (2005): Contains 1178 protein structures with the goal of classifying enzymes vs. non enzymes. GCN outperforms all other models on this dataset, however the Baseline model, where no structure is used also performs very similarly. This suggests that the graph structure within this dataset does not add much information over the structure encoded in the eccentricity and clustering coefficient.

PTC Toivonen et al. (2003): Contains 344 chemical compound graphs divided into two classes based on whether or not they cause cancer in rats. This dataset is very difficult to classify without features however LEGS-RBF and LEGS-FCN are able to capture the long range connections slightly better than other methods.

COLLAB Yanardag & Vishwanathan (2015): 5000 ego-networks of different researchers from high energy physics, condensed matter physics or astrophysics. The goal is to determine which field the research belongs to. The GraphSAGE model performs best on this dataset although the LEGS-RBF network performs nearly as well. Ego graphs have a very small average diameter. Thus shallow networks can perform quite well on them as is the case here.

IMDB Yanardag & Vishwanathan (2015): For each graph nodes represent actresses/actors and there is an edge between them if they are in the same move. These graphs are also ego graphs around specific actors. IMDB-BINARY classifies between action and romance genres. IMDB-MULTI classifies between 3 classes. Somewhat surprisingly GS-SVM performs the best with other LEGS networks close behind. This could be due to oversmoothing on the part of GCN and GraphSAGE when the graphs are so small.

REDDIT Yanardag & Vishwanathan (2015): Graphs in REDDIT-BINARY/MULTI-5K/MULTI-12K datasets each graph represents a discussion thread where nodes correspond to users and there is an edge between two nodes if one replied to the other's comment. The task is to identify which subreddit a given graph came from. On these datasets GCN outperforms other models.

Table S1: Dataset statistics, diameter, nodes, edges, clustering coefficient averaged over all graphs. Split into bio-chemical and social network types.

	# Graphs	# Classes	Diameter	Nodes	Edges	Clust. Coeff
DD	1178	2	19.81	284.32	715.66	0.48
ENZYMES	600	6	10.92	32.63	62.14	0.45
MUTAG	188	2	8.22	17.93	19.79	0.00
NCI1	4110	2	13.33	29.87	32.30	0.00
NCI109	4127	2	13.14	29.68	32.13	0.00
PROTEINS	1113	2	11.62	39.06	72.82	0.51
PTC	344	2	7.52	14.29	14.69	0.01
COLLAB	5000	3	1.86	74.49	2457.22	0.89
IMDB-BINARY	1000	2	1.86	19.77	96.53	0.95
IMDB-MULTI	1500	3	1.47	13.00	65.94	0.97
REDDIT-BINARY	2000	2	8.59	429.63	497.75	0.05
REDDIT-MULTI-12K	11929	11	9.53	391.41	456.89	0.03
REDDIT-MULTI-5K	4999	5	10.57	508.52	594.87	0.03

Table S2: Mean $\pm$ std. over 10 test sets on bio-chemical and social datasets.

	LEGS-RBF	LEGS-FCN	LEGS-FIXED	GCN	GraphSAGE	GS-SVM	Baseline
DD	72.58 $\pm$ 3.35	72.07 $\pm$ 2.37	69.09 $\pm$ 4.82	67.82 $\pm$ 3.81	66.37 $\pm$ 4.45	72.66 $\pm$ 4.94	**75.98 $\pm$ 2.81**
ENZYMES	36.33 $\pm$ 4.50	**38.50 $\pm$ 8.18**	32.33 $\pm$ 5.04	31.33 $\pm$ 6.89	15.83 $\pm$ 9.10	27.33 $\pm$ 5.10	20.50 $\pm$ 5.99
MUTAG	33.51 $\pm$ 4.34	82.98 $\pm$ 9.85	81.84 $\pm$ 11.24	79.30 $\pm$ 9.66	81.43 $\pm$ 11.64	**85.09 $\pm$ 7.44**	79.80 $\pm$ 9.92
NCI1	**74.26 $\pm$ 1.53**	70.83 $\pm$ 2.65	71.24 $\pm$ 1.63	60.80 $\pm$ 4.26	57.54 $\pm$ 3.33	69.68 $\pm$ 2.38	56.69 $\pm$ 3.07
NCI109	**72.47 $\pm$ 2.11**	70.17 $\pm$ 1.46	69.25 $\pm$ 1.75	61.30 $\pm$ 2.99	55.15 $\pm$ 2.58	68.55 $\pm$ 2.06	57.38 $\pm$ 2.20
PROTEINS	70.89 $\pm$ 3.91	71.06 $\pm$ 3.17	67.30 $\pm$ 2.94	**74.03 $\pm$ 3.20**	71.87 $\pm$ 3.50	70.98 $\pm$ 2.67	73.22 $\pm$ 3.76
PTC	**57.26 $\pm$ 5.54**	56.92 $\pm$ 9.36	54.31 $\pm$ 6.92	56.34 $\pm$ 10.29	55.22 $\pm$ 9.13	56.96 $\pm$ 7.09	56.71 $\pm$ 5.54
COLLAB	75.78 $\pm$ 1.95	75.40 $\pm$ 1.80	72.94 $\pm$ 1.70	73.80 $\pm$ 1.73	**76.12 $\pm$ 1.58**	74.54 $\pm$ 2.32	64.76 $\pm$ 2.63
IMDB-BINARY	64.90 $\pm$ 3.48	64.50 $\pm$ 3.50	64.30 $\pm$ 3.68	47.40 $\pm$ 6.24	46.40 $\pm$ 4.03	**66.70 $\pm$ 3.53**	47.20 $\pm$ 5.67
IMDB-MULTI	41.93 $\pm$ 3.01	40.13 $\pm$ 2.77	41.67 $\pm$ 3.19	39.33 $\pm$ 3.13	39.73 $\pm$ 3.45	**42.13 $\pm$ 2.53**	39.53 $\pm$ 3.63
REDDIT-BINARY	**86.10 $\pm$ 2.92**	78.15 $\pm$ 5.42	85.00 $\pm$ 1.93	81.60 $\pm$ 2.32	73.40 $\pm$ 4.38	85.15 $\pm$ 2.78	69.30 $\pm$ 5.08
REDDIT-MULTI-12K	38.47 $\pm$ 1.07	38.46 $\pm$ 1.31	39.74 $\pm$ 1.31	**42.57 $\pm$ 0.90**	32.17 $\pm$ 2.04	39.79 $\pm$ 1.11	22.07 $\pm$ 0.98
REDDIT-MULTI-5K	47.83 $\pm$ 2.61	46.97 $\pm$ 3.06	47.17 $\pm$ 2.93	**52.79 $\pm$ 2.11**	45.71 $\pm$ 2.88	48.79 $\pm$ 2.95	36.41 $\pm$ 1.80

Table S3: Mean $\pm$ std. over test set selection on cross-validated LEGS-RBF Net with reduced training set size.

Train, Val, Test %	80%, 10%, 10%	70%, 10%, 20%	40%, 10%, 50%	20%, 10%, 70%
COLLAB	75.78 $\pm$ 1.95	75.00 $\pm$ 1.83	74.00 $\pm$ 0.51	72.73 $\pm$ 0.59
DD	72.58 $\pm$ 3.35	70.88 $\pm$ 2.83	69.95 $\pm$ 1.85	69.43 $\pm$ 1.24
ENZYMES	36.33 $\pm$ 4.50	34.17 $\pm$ 3.77	29.83 $\pm$ 3.54	23.98 $\pm$ 3.32
IMDB-BINARY	64.90 $\pm$ 3.48	63.00 $\pm$ 2.03	63.30 $\pm$ 1.27	57.67 $\pm$ 6.04
IMDB-MULTI	41.93 $\pm$ 3.01	40.80 $\pm$ 1.79	41.80 $\pm$ 1.23	36.83 $\pm$ 3.31
MUTAG	33.51 $\pm$ 4.34	33.51 $\pm$ 1.14	33.52 $\pm$ 1.26	33.51 $\pm$ 0.77
NCI1	74.26 $\pm$ 1.53	74.38 $\pm$ 1.38	72.07 $\pm$ 0.28	70.30 $\pm$ 0.72
NCI109	72.47 $\pm$ 2.11	72.21 $\pm$ 0.92	70.44 $\pm$ 0.78	68.46 $\pm$ 0.96
PROTIENS	70.89 $\pm$ 3.91	69.27 $\pm$ 1.95	69.72 $\pm$ 0.27	68.96 $\pm$ 1.63
PTC	57.26 $\pm$ 5.54	57.83 $\pm$ 4.39	54.62 $\pm$ 3.21	55.45 $\pm$ 2.35
REDDIT-BINARY	86.10 $\pm$ 2.92	86.05 $\pm$ 2.51	85.15 $\pm$ 1.77	83.71 $\pm$ 0.97
REDDIT-MULTI-12K	38.47 $\pm$ 1.07	38.60 $\pm$ 0.52	37.55 $\pm$ 0.05	36.65 $\pm$ 0.50
REDDIT-MULTI-5K	47.83 $\pm$ 2.61	47.81 $\pm$ 1.32	46.73 $\pm$ 1.46	44.59 $\pm$ 1.02

Table S4: Test set mean squared error on CASP GDT regression task across targets over 3 non-overlapping test sets.

	LEGS-RBF	LEGS-FCN	LEGS-FIXED	GCN	GraphSAGE	Baseline
t0860	197.68 $\pm$ 34.29	**164.22 $\pm$ 10.28**	206.20 $\pm$ 28.46	314.90 $\pm$ 29.66	230.45 $\pm$ 79.72	414.41 $\pm$ 26.96
t0868	131.42 $\pm$ 8.12	**127.71 $\pm$ 14.26**	178.45 $\pm$ 5.64	272.14 $\pm$ 26.34	191.08 $\pm$ 21.96	411.98 $\pm$ 57.39
t0869	106.69 $\pm$ 9.97	132.12 $\pm$ 31.37	**104.47 $\pm$ 14.16**	317.22 $\pm$ 12.75	244.38 $\pm$ 40.58	393.12 $\pm$ 48.70
t0872	144.11 $\pm$ 24.88	148.20 $\pm$ 23.63	**134.48 $\pm$ 8.25**	293.96 $\pm$ 19.00	221.13 $\pm$ 28.74	374.48 $\pm$ 33.70
t0879	89.00 $\pm$ 44.94	80.14 $\pm$ 16.21	**64.63 $\pm$ 15.92**	309.23 $\pm$ 69.40	172.41 $\pm$ 73.07	364.79 $\pm$ 144.32
t0900	193.74 $\pm$ 10.78	171.05 $\pm$ 25.41	**158.56 $\pm$ 9.87**	254.11 $\pm$ 18.63	209.07 $\pm$ 11.90	399.16 $\pm$ 83.48
t0912	**113.00 $\pm$ 22.31**	169.55 $\pm$ 27.35	150.70 $\pm$ 8.53	227.17 $\pm$ 22.11	192.28 $\pm$ 39.45	406.25 $\pm$ 31.42
t0920	**80.46 $\pm$ 14.98**	136.94 $\pm$ 36.43	84.83 $\pm$ 19.70	361.19 $\pm$ 71.25	261.72 $\pm$ 59.67	398.22 $\pm$ 25.60
t0921	187.89 $\pm$ 46.15	165.97 $\pm$ 42.39	**142.97 $\pm$ 27.09**	382.69 $\pm$ 20.27	260.49 $\pm$ 16.09	363.92 $\pm$ 35.79
t0922	254.83 $\pm$ 91.28	**110.54 $\pm$ 43.99**	227.73 $\pm$ 26.41	366.72 $\pm$ 8.10	290.71 $\pm$ 7.22	419.14 $\pm$ 45.49
t0942	188.55 $\pm$ 11.10	167.53 $\pm$ 22.01	**137.21 $\pm$ 7.43**	371.31 $\pm$ 9.90	233.78 $\pm$ 84.95	393.03 $\pm$ 24.93
t0944	146.59 $\pm$ 8.41	**138.67 $\pm$ 50.36**	245.79 $\pm$ 58.16	263.03 $\pm$ 9.43	199.40 $\pm$ 51.11	404.12 $\pm$ 40.82

C TRAINING DETAILS

We train all models for a maximum of 1000 epochs with an initial learning rate of $1e^{-4}$ using the ADAM optimizer (Kingma & Ba, 2015). We terminate training if validation loss does not improve for 100 epochs testing every 10 epochs. Our models are implemented with Pytorch Paszke et al. (2019) and Pytorch geometric. Models were run on a variety of hardware resources. For all models we use $q = 4$ normalized statistical moments for the node to graph level feature extraction and $m = 16$ diffusion scales in line with choices in Gao et al. (2019).

Table S5: Quantified distance between the empirically observed enzyme class exchange preferences of Cuesta et al. (2015) and the class exchange preferences inferred from LEGS-FIXED, LEGS-FCN, and a GCN. We measure the cosine distance between the graphs represented by the chord diagrams in Figure 2. As before, the self-affinities were discarded. LEGS-Fixed reproduces the exchange preferences the best, but LEGS-FCN still reproduces well and has significantly better classification accuracy.

LEGS-FIXED	LEGS-FCN	GCN
0.132	0.146	0.155

C.1 Cross Validation Procedure

For all datasets we use 10-fold cross validation with 80% training data 10% validation data and 10% test data for each model. We first split the data into 10 (roughly) equal partitions. For each model we take exactly one of the partitions to be the test set and one of the remaining nine to be the validation set. We then train the model on the remaining eight partitions using the cross-entropy loss on the validation for early stopping checking every ten epochs. For each test set, we use majority voting of the nine models trained with that test set. We then take the mean and standard deviation across these test set scores to average out any variability in the particular split chosen. This results in 900 models trained on every dataset. With mean and standard deviation over 10 ensembled models each with a separate test set.