Learnable Filters for Geometric Scattering Modules

Alexander Tong, Frederik Wenkel, Dhananjay Bhaskar, *Member, IEEE*, Kincaid Macdonald, Jackson Grady, Michael Perlmutter, Smita Krishnaswamy, *Member, IEEE*, and Guy Wolf

***Abstract*—We propose a new graph neural network (GNN) module, based on relaxations of recently proposed geometric scattering transforms, which consist of a cascade of graph wavelet filters. Our learnable geometric scattering (LEGS) module enables adaptive tuning of the wavelets to encourage band-pass features to emerge in learned representations. The incorporation of our LEGS-module in GNNs enables the learning of longer-range graph relations compared to many popular GNNs, which often rely on encoding graph structure via smoothness or similarity between neighbors. Further, its wavelet priors result in simplified architectures with significantly fewer learned parameters compared to competing GNNs. We demonstrate the predictive performance of LEGS-based networks on graph classification benchmarks, as well as the descriptive quality of their learned features in biochemical graph data exploration tasks. Our results show that LEGS-based networks match or outperforms popular GNNs, as well as the original geometric scattering construction, on many datasets, in particular in biochemical domains, while retaining certain mathematical properties of handcrafted (non-learned) geometric scattering.**

***Index Terms*—Geometric scattering, graph neural networks, graph signal processing.**

Manuscript received 27 January 2023; revised 20 December 2023 and 10 February 2024; accepted 29 February 2024. Date of publication 18 March 2024; date of current version 24 June 2024. The work of Frederik Wenkel was supported by the Fin-ML CREATE Graduate Studies Scholarship for Ph.D., J.A. DeSève Scholarship for Ph.D. The work of Dhananjay Bhaskar was supported by Yale-Boehringer Ingelheim Biomedical Data Science Fellowship. The work of Smita Krishnaswamy was supported in part by Chan-Zuckerberg Initiative under Grant 182702 and Grant CZF2019-002440, in part by the NSF Career under Grant 2047856, and in part by the Sloan Fellowship under Grant FG-2021-15883. The work of Guy Wolf was supported in part by CIFAR AI Chair, in part by the IVADO under Grant PRF-2019-3583139727, in part by the FRQNT under Grant 299376, and in part by the NSERC Discovery under Grant 03267. The work of Michael Perlmutter, Smita Krishnaswamy, and Guy Wolf was supported in part by the NSF under Grant 2327211. The work of Smita Krishnaswamy and Guy Wolf was supported in part by the NIH under Grant R01GM135929 and Grant R01GM130847. The associate editor coordinating the review of this manuscript and approving it for publication was Ms. Irène Waldspurger. *(Alexander Tong and Frederik Wenkel are co-first authors; Smita Krishnasmamy and Guy Wolf are co-last authors.) (Corresponding author: Smita Krishnaswamy.)*

Alexander Tong is with the Department of Computer Science and Operations Research, Université de Montréal, and also with Mila – the Quebec AI Institute, Montreal, QC H2S 3H1, Canada.

Frederik Wenkel and Guy Wolf are with the Department of Mathematics and Statistics, Université de Montréal, and also with Mila – the Quebec AI Institute, Montreal, QC H2S 3H1, Canada.

Dhananjay Bhaskar is with the Department of Genetics, Yale University, New Haven, CT 06520 USA.

Kincaid Macdonald is with the Department of Mathematics, Yale University, New Haven, CT 06520 USA.

Jackson Grady is with the Department of Computer Science, Yale University, New Haven, CT 06520 USA.

Michael Perlmutter was with the Department of Mathematics, University of California 90048, Los Angeles. He is now with the Department of Mathematics, Boise State University, Boise, ID 83725 USA.

Smita Krishnaswamy is with the Department of Genetics and Department of Computer Science, Yale University, New Haven, CT 06520 USA (e-mail: smita.krishnaswamy@yale.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TSP.2024.3378001, provided by the authors.

Digital Object Identifier 10.1109/TSP.2024.3378001

I. Introduction

GEOMETRIC deep learning has recently emerged as an increasingly prominent branch of deep learning [1]. 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 [2], [3], [4], [5], [6], [7]. However, recent work has shown the difficulty in generalizing these methods to more complex structures, identifying common problems and phrasing them in terms of so-called oversmoothing [8], underreaching [9] and oversquashing [10].

Using graph signal processing terminology from [4], these issues can be partly attributed to the limited construction of convolutional filters in many commonly used GCN architectures. Inspired by the filters learned in convolutional neural networks, GCNs consider node features as graph signals and aim to aggregate information from neighboring nodes. For example, [4] presented a typical implementation of a GCN with a cascade of averaging (essentially low pass) filters. We note that more general variations of GCN architectures exist [3], [5], [6], which are capable of representing other filters, but as investigated in [10], they often have difficulty in learning long-range connections.

Recently, an alternative approach was presented to provide deep geometric representation learning by generalizing Mallat's scattering transform [11], originally proposed to provide a mathematical framework for understanding convolutional neural networks, to graphs [12], [13], [14] and manifolds [15], [16], [17]. The geometric scattering transform can represent nodes or graphs based on multi-scale diffusions, and differences between scales of diffusions of graph signals (i.e., node features). Similar to traditional scattering, which can be seen as a convolutional network with non-learned wavelet filters, geometric scattering is defined as a GNN with handcrafted graph filters, constructed with diffusion wavelets over the input graph [18], which are then cascaded with pointwise absolute-value nonlinearities. The efficacy of geometric scattering features in

graph processing tasks was demonstrated in [12], 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 [19], [20].

However, while geometric scattering transforms provide effective universal feature extractors, their handcrafted design does not allow the automatic task-driven representation learning that is so successful in traditional GNNs and neural networks in general. Here, we combine both frameworks by incorporating richer multi-frequency band features from geometric scattering into GNNs, while allowing them to be flexible and trainable. We introduce the geometric scattering module, which can be used within a larger neural network. We call this a *learnable geometric scattering (LEGS) module* and show it inherits properties from the scattering transform while allowing the scales of the diffusion to be learned. Moreover, we show that our framework is differentiable, allowing for backpropagation through it in a standard reverse mode auto differentiation library.

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. VI. In particular, we find that our network maintains the robustness to small training sets present in fixed geometric scattering [21], while improving performance on biological graph classification and regression tasks. In particular, we find that in tasks where the graphs have a large diameter relative to their size, learnable scattering features improve performance over competing methods. We show that our construction performs better on tasks that require 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. We also show that our network maintains performance in social network and other node classification tasks where state-of-the-art GNNs perform well.

A previous short version of this work appeared in the IEEE Workshop on Machine Learning and Signal Processing 2021 [22]. We expand on that work first by incorporating additional theory including Theorem 3 which generalizes existing theory for nonexpansive scattering operators. Furthermore, we add additional experiments on molecular data, as well as ablation studies on both amount of training data and ensembling with other models.

The remainder of this paper is organized as follows. In Section II, we review related work on graph scattering and literature on the challenges of modern GNNs. In Section III, we review some of the concepts of geometric scattering. In Section IV, we present expanded theory on geometric scattering with task-driven tuning. This theory establishes that our LEGS module retains the theoretical properties of scattering while increasing expressiveness. In Section V, we present the architecture and implementation details of the LEGS module. We examine the empirical performance of LEGS architectures in Section VI and conclude in Section VII.

II. Related Work

A widely discussed challenge for many modern GNN approaches is so-called oversmoothing [8], [23]. This is a result of the classic message passing in GNNs that is based on cascades of local node feature aggregations over node neighborhoods. This increasingly smooths graph signals, which in turn renders the graph nodes undistinguishable. From a spectral point of view, this phenomenon is due to most GNN filters being low-pass filters [24] that mostly preserve the low-frequency spectrum. A related phenomenon is so-called underreaching [9], which is a result of the limited spatial support of most GNN architectures. Most models can only relate information from nodes within a distance equal to the number of layers. Hence, they cannot represent long-range interactions as the before mentioned oversmoothing typically prohibits the design of truly "deep" GNN architectures. Lastly, oversquashing [10] is yet another consequence of typical message passing. As the number of nodes in the receptive field of each node grows exponentially in the number of GNN layers, a huge amount of information needs to be compressed into a vector of fixed size. This makes it difficult to represent meaningful relationships between nodes, as the contribution of features of single nodes becomes marginal.

We note that efforts to improve the capture of long-range connections in graph representation learning have recently yielded several spectral approaches based on using the Lancoz algorithm to approximate graph spectra [25], or based on learning in block Krylov subspaces [26]. Such methods are complementary to the work presented here, in that their spectral approximation can also be applied in the computation of geometric scattering when considering very long range scales (e.g., via spectral formulation of graph wavelet filters). However, we find that such approximations are not necessary in the datasets considered here and in other recent work focusing on whole-graph tasks, where direct computation of polynomials of the Laplacian is sufficient. Furthermore, recent attempts have also considered ensemble approaches with hybrid architectures that combine GCN and scattering channels [27], albeit primarily focused on node-level tasks, considered on a single graph at a time, rather than whole-graph tasks considered here on datasets comparing multiple graphs. Such ensemble approaches are also complimentary to the proposed approach in that hybrid architectures can also be applied in conjunction with the proposed LEGS module here as we demonstrate in Sec. VI.

III. Preliminaries: Geometric Scattering

Let $\mathcal{G} = (V, E, w)$ be a weighted graph with $V := \{v_1, \ldots, v_n\}$ the set of nodes, $E \subset \{\{v_i, v_j\} \in \binom{V}{2}, i \neq j\}$ the set of (undirected) edges and $w : E \to (0, \infty)$ assigning (positive) edge weights to the graph edges. 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 $x \in \mathbb{R}^n$ with the i^{th} entry being $x(v_i)$. We define the *weighted adjacency matrix* $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ of $\mathcal{G}$ as $\boldsymbol{W}[v_i, v_j] := w(v_i, v_j)$ if $\{v_i, v_j\} \in E$, and 0 otherwise and the *degree matrix* $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ of $\mathcal{G}$ as $\boldsymbol{D} := \mathrm{diag}(d_1, \ldots, d_n)$ with $d_i := \deg(v_i) := \sum_{j=1}^{n} \boldsymbol{W}[v_i, v_j]$ the *degree* of node v_i.

The geometric scattering transform [12] consists of a cascade of graph filters constructed from a left stochastic diffusion matrix $\boldsymbol{P} := \frac{1}{2}\left(\boldsymbol{I}_n + \boldsymbol{W}\boldsymbol{D}^{-1}\right)$, 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 staying at the current node or transitioning to a neighbor. Scattering filters are defined via graph-wavelet matrices $\boldsymbol{\Psi}_j \in \mathbb{R}^{n\times n}$ of order $j \in \mathbb{N}_0$, as

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

These diffusion wavelet operators partition the frequency spectrum into dyadic frequency bands, which are then organized into a full wavelet filter bank $\mathcal{W}_J := \{\boldsymbol{\Psi}_j, \boldsymbol{\Phi}_J\}_{0\leq j\leq J}$, where $\boldsymbol{\Phi}_J := \boldsymbol{P}^{2^J}$ 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 [20], 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 such as those considered in [13].

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

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

parameterized by the scattering path $p := (j_1, \ldots, j_m) \in \cup_{m\in\mathbb{N}}\mathbb{N}_0^m$ that determines the filter scales of each wavelet. Whole-graph representations are obtained by aggregating node-level features via statistical moments over the nodes of the graph [12], which yields the geometric scattering moments

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

indexed by the scattering path p and moment order q. Notably, coefficients with $m = 0$ correspond to simply taking moments of the input signal $\mathbf{x}$ and, the coefficients with $m = 1$ correspond to moments of the wavelet transform. For $m \geq 2$, the coefficients correspond to iterated wavelet transforms with non-linearities interspersed. Finally, we note that Theorem 3.6 of [20] shows that $\boldsymbol{U}_p$ is equivariant to permutations of the nodes. Therefore, it follows that the graph-level scattering transform $\boldsymbol{S}_{p,q}$ is node-permutation invariant since it is defined via global summation.

IV. Adaptive Geom. Scattering Relaxation

The geometric scattering construction, described in Sec. III, can be seen as a particular GNN architecture with handcrafted layers, rather than learned ones. This provides a solid mathematical framework for understanding the encoding of geometric information in GNNs [20], while also providing effective unsupervised graph representation learning for data exploration, which also has advantages in supervised learning tasks [12].

Both [20] and [12] used dyadic scales in Eq. 1, a choice inspired by the Euclidean scattering transform [11]. Below in Theorem 1, we will show that these dyadic scales may be replaced by *any* increasing sequence of scales and the resulting wavelets will still form a nonexpansive frame. Later in Section V, Theorem 3 will consider scales which are learned from data and show that the learned filter bank forms a nonexpansive operator under mild assumptions. This allows us to obtain a flexible model with similar guarantees to the model considered in [12], but which is amenable to task-driven tuning provided by end-to-end GNN training.

Given an increasing sequence of integer diffusion time scales $0 < t_1 < \cdots < t_J$, we replace the wavelets considered in Eq. 1 with the generalized filter bank $\mathcal{W}'_J := \{\boldsymbol{\Psi}'_j, \boldsymbol{\Phi}'_J\}_{j=0}^{J-1}$, where

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

Since $\boldsymbol{P}$ is not a symmetric matrix, the these wavelets are not self-adjoint operators on the standard, unweighted inner product space. Therefore, to study these wavelets, it will be convenient to consider a weighted inner product space $L^2(\mathcal{G}, \boldsymbol{D}^{-\frac{1}{2}})$ of graph signals with inner product $\langle \boldsymbol{x}, \boldsymbol{y}\rangle := \langle \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{x}, \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{y}\rangle$ and induced norm $\|\boldsymbol{x}\|^2_{\boldsymbol{D}^{-\frac{1}{2}}} = \|\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{x}\|_2 = \sum_{i=1}^{n}\frac{\boldsymbol{x}[i]^2}{d_i}$ for $\boldsymbol{x}, \boldsymbol{y} \in L^2(\mathcal{G}, \boldsymbol{D}^{-\frac{1}{2}})$. Importantly, we note that Lemma 1.1 of [15] implies that $\boldsymbol{P}$ (and therefore the wavelets constructed from it) are self-adjoint on $L^2(\mathcal{G}, \boldsymbol{D}^{-\frac{1}{2}})$.

The following theorem shows that $\mathcal{W}'_J$ is a nonexpansive frame, similar to the result shown for dyadic scales in [20]. We note our proof that it relies upon the fact that $\boldsymbol{D}^{-1/2}\boldsymbol{P}\boldsymbol{D}^{1/2}$ is a symmetric matrix, which necessitates that our results be in terms of the weighted norm $\|\cdot\|_{\boldsymbol{D}^{-1/2}}$ rather than the standard unweighted inner product.

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

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

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

Proof: Note that $\boldsymbol{P}$ has a symmetric conjugate $\boldsymbol{M} := \boldsymbol{D}^{-1/2}\boldsymbol{P}\boldsymbol{D}^{1/2}$ with eigendecomposition $\boldsymbol{M} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top$ for orthogonal $\boldsymbol{Q}$. Given this decomposition, we can write

$$\begin{aligned}\boldsymbol{\Phi}'_J &= \boldsymbol{D}^{1/2}\boldsymbol{Q}\boldsymbol{\Lambda}^{t_J}\boldsymbol{Q}^\top\boldsymbol{D}^{-1/2},\\ \boldsymbol{\Psi}'_j &= \boldsymbol{D}^{1/2}\boldsymbol{Q}(\boldsymbol{\Lambda}^{t_j} - \boldsymbol{\Lambda}^{t_{j+1}})\boldsymbol{Q}^\top\boldsymbol{D}^{-1/2}, \quad 0 \leq j \leq J-1,\end{aligned}$$

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

$$\|\boldsymbol{\Phi}'_J\boldsymbol{x}\|^2_{\boldsymbol{D}^{-1/2}} = \langle \boldsymbol{\Phi}'_J\boldsymbol{x}, \boldsymbol{\Phi}'_J\boldsymbol{x}\rangle_{\boldsymbol{D}^{-1/2}} = \|\boldsymbol{\Lambda}^{t_J}\boldsymbol{Q}^\top\boldsymbol{D}^{-1/2}\boldsymbol{x}\|_2^2.$$

If we consider a change of variable to $\boldsymbol{y} = \boldsymbol{Q}^\top\boldsymbol{D}^{-1/2}\boldsymbol{x}$, we have $\|\boldsymbol{x}\|^2_{\boldsymbol{D}^{-1/2}} = \|\boldsymbol{D}^{-1/2}\boldsymbol{x}\|_2^2 = \|\boldsymbol{y}\|_2^2$, while $\|\boldsymbol{\Phi}'_J\boldsymbol{x}\|^2_{\boldsymbol{D}^{-1/2}} = \|\boldsymbol{\Lambda}^{t_J}\boldsymbol{y}\|_2^2$. Similarly, we can also reformulate the operations of the other filters in terms of diagonal matrices applied to $\boldsymbol{y}$ as $\|\boldsymbol{\Psi}'_j\boldsymbol{x}\|^2_{\boldsymbol{D}^{-1/2}} = \|(\boldsymbol{\Lambda}^{t_j} - \boldsymbol{\Lambda}^{t_{j+1}})\boldsymbol{y}\|_2^2$.

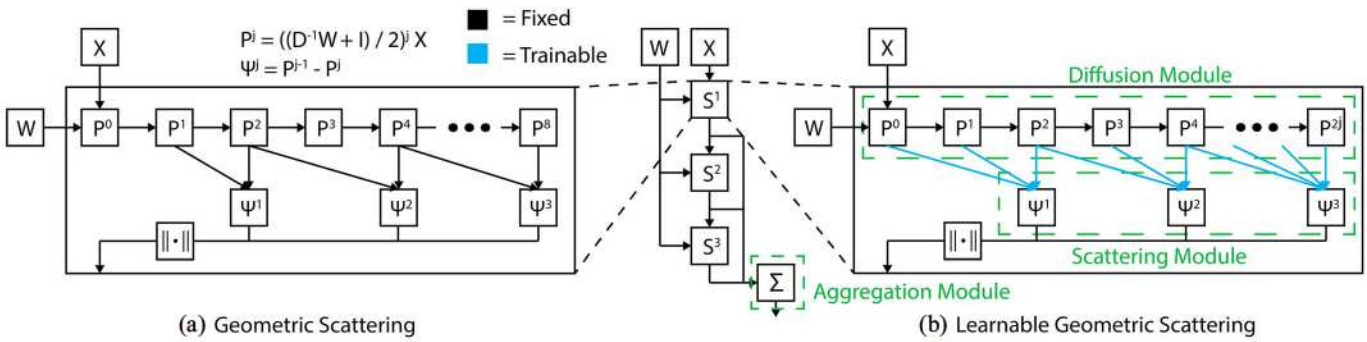


Fig. 1. LEGS module learns to select the appropriate scattering scales from the data.

Given these reformulations, we can now write

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

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

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

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 $t_1 > 0 = t_0$,

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

and thus, setting $C := \min_{0\leq\xi\leq 1}(\xi^{2t_J} + (1-\xi^{t_1})^2) > 0$, we get the lower bound $\|\Lambda^{t_J}\boldsymbol{y}\|_2^2 + \sum_{j=0}^{J-1}\|(\Lambda^{t_j} - \Lambda^{t_{j+1}})\boldsymbol{y}\|_2^2 \geq C\|\boldsymbol{y}\|_2^2$. Applying the reverse change of variable to $\boldsymbol{x}$ and $L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$ yields the result of the theorem. □

Theorem 1 shows that the wavelet transform is injective and stable to additive noise. Our next theorem shows that it is permutation equivariant at the node-level and permutation invariant at the graph level. This guarantees that the extracted features encode the intrinsic geometry of the graph rather than a priori indexation.

Theorem 2: Let $\boldsymbol{U}'_p$ and $\boldsymbol{S}'_{p,q}$ be defined as in Eq. 2-3, with the filters from $\mathcal{W}'_J$ with an arbitrary configuration $0 < t_1 < \cdots < t_J$ in place of $\mathcal{W}_J$. Then, for any permutation Π over the nodes and any graph signal $\boldsymbol{x} \in L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$, we have $\boldsymbol{U}'_p\Pi\boldsymbol{x} = \Pi\boldsymbol{U}'_p\boldsymbol{x}$ and $\boldsymbol{S}'_{p,q}\Pi\boldsymbol{x} = \boldsymbol{S}'_{p,q}\boldsymbol{x}$, for $p \in \cup_{m\in\mathbb{N}}\mathbb{N}_0^m$, $q \in \mathbb{N}$, where geometric scattering implicitly considers the node ordering supporting its input signal.

Proof: For any permutation Π, we let $\overline{\mathcal{G}} = \Pi(\mathcal{G})$ be the graph obtained by permuting the vertices of $\mathcal{G}$ with Π. We note that the adjacency and degree matrices on $\overline{\mathcal{G}}$ are given by $\overline{\boldsymbol{W}} = \Pi\boldsymbol{W}\Pi^\top$ and $\overline{\boldsymbol{D}} = \Pi\boldsymbol{D}\Pi^\top$. Additionally, the corresponding permutation operation on a graph signal $\boldsymbol{x} \in L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$ gives a signal $\Pi\boldsymbol{x} \in L^2(\overline{\mathcal{G}}, \overline{\boldsymbol{D}}^{-1/2})$, which we implicitly considered in the statement of the theorem, without specifying these notations for simplicity. Rewriting the statement of the theorem more formally, we let $\overline{\boldsymbol{P}} = \frac{1}{2}(\boldsymbol{I}_n + \overline{\boldsymbol{W}}\overline{\boldsymbol{D}}^{-1})$ denote the analog of $\boldsymbol{P}$ on $\overline{\mathcal{G}}$ and let $\overline{\boldsymbol{\Psi}}'_j$ denote the matrix defined according to Eq. 4 but with $\overline{\boldsymbol{P}}$ in place of $\boldsymbol{P}$. Then, we let $\overline{\boldsymbol{U}}'_p$ and $\overline{\boldsymbol{S}}'_{p,q}$ be the analogs of $\boldsymbol{U}'_p$ and $\boldsymbol{S}'_{p,q}$ constructed using $\overline{\boldsymbol{\Psi}}'_j$. With the introduced notations, we aim to show that $\overline{\boldsymbol{U}}'_p\Pi\boldsymbol{x} = \Pi\boldsymbol{U}'_p\boldsymbol{x}$ and $\overline{\boldsymbol{S}}'_{p,q}\Pi\boldsymbol{x} = \boldsymbol{S}'_{p,q}\boldsymbol{x}$.

First, we notice that for any $\boldsymbol{\Psi}_j$, $0 < j < J$ we have that $\overline{\boldsymbol{\Psi}}_j\Pi\boldsymbol{x} = \Pi\boldsymbol{\Psi}_j\boldsymbol{x}$, as for $1 \leq j \leq J-1$,

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

with similar reasoning for $j \in \{0, J\}$. Note that the element-wise absolute value yields $|\Pi\boldsymbol{x}| = \Pi|\boldsymbol{x}|$ for any permutation matrix Π. These two observations inductively yield

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

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

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

which completes the proof of the theorem. □

We note that the results in Theorems 1-2 and their proofs closely follow the theoretical framework proposed by [20]. We carefully account here for the relaxed learned configuration, which replaces the original handcrafted one there. We also note that conclusions of Theorem 2 remain valid if $|\cdot|$ is replaced with any vertex-wise nonlinearity.

V. A Learnable Geom. Scattering Module

In this section, we show how the generalized geometric scattering construction presented in Sec. IV can be implemented in a data-driven way via a backpropagation-trainable module. Throughout this section, we consider an input graph signal $\boldsymbol{x} \in \mathbb{R}^n$ or, equivalently, a collection of graph signals $\boldsymbol{X} \in \mathbb{R}^{n\times N_{\ell-1}}$. For simplicity, our theory will focus on the case where there is a single signal x. However, the numerical implementation proceeds in the exact same manner with multiple

signals and our theoretical results may also be easily adapted to the multiple signal case. The forward propagation of these signals can be divided into three major submodules. First, a *diffusion submodule* implements the Markov process that forms the basis of the filter bank and transform. Then, a *scattering submodule* 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.

The diffusion submodule. We build a set of $m \in \mathbb{N}$ subsequent diffusion steps of the signal x by iteratively multiplying the diffusion matrix P to the left of the signal, resulting in $\left[Px, P^2x, \ldots, P^{t_{\max}}x\right]$. Since P is often sparse, for efficiency reasons these filter responses are implemented via an RNN structure consisting of $t_{\max}$ RNN modules. Each module propagates the incoming hidden state $h_{t-1}, t = 1, \ldots, t_{\max}$ with P with the readout o_t equal to the produced hidden state, $h_t := Ph_{t-1}, \quad o_t := h_t$.

The scattering submodule. Next, we consider the selection of $J \leq m$ diffusion scales for the flexible filter bank construction with 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. III and Eq. 1), fixed for all datasets (LEGS-FIXED); and 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 scale selection, we use a selection matrix $F \in \mathbb{R}^{J \times t_{\max}}$, where each row $F_{(j,\cdot)}, j = 1, \ldots, J$ is dedicated to identifying the diffusion scale of the wavelet P^{t_j} via a one-hot encoding. This is achieved by setting $F := \sigma(\Theta) = [\sigma(\theta_1), \sigma(\theta_2), \ldots, \sigma(\theta_J)]^\top$, where $\theta_j \in \mathbb{R}^{t_{\max}}$ constitute the rows of the trainable weight matrix Θ, and σ is the softmax function (see [28], section 6.2.2.3). For $\theta = [\theta_1, \ldots, \theta_{t_{\max}}] \in \mathbb{R}^{t_{\max}}$ and $\sigma(\theta)_i := e^{\theta_i} / \sum_j e^{\theta_j}$, we have

$$F = \begin{bmatrix} \sigma(\theta_1)_1 & \ldots & \sigma(\theta_1)_{t_{\max}} \\ \sigma(\theta_2)_1 & \ldots & \sigma(\theta_2)_{t_{\max}} \\ \vdots & \ddots & \vdots \\ \sigma(\theta_J)_1 & \ldots & \sigma(\theta_J)_{t_{\max}} \end{bmatrix}.$$

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 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 $\widetilde{\mathcal{W}}_F := \{\widetilde{\Psi}_j, \widetilde{\Phi}_J\}_{j=0}^{J-1}$ with the filters

$$\widetilde{\Psi}_0 x = x - \sum_{t=1}^{t_{\max}} F_{(1,t)} P^t x, \quad \widetilde{\Phi}_J x = \sum_{t=1}^{t_{\max}} F_{(J,t)} P^t x,$$
$$\widetilde{\Psi}_j x = \sum_{t=1}^{t_{\max}} \left[F_{(j,t)} P^t x - F_{(j+1,t)} P^t x\right], \quad 1 \leq j \leq J-1, \tag{5}$$

matching and implementing the construction of $\mathcal{W}'_J$ (Eq. 4). We further illustrate the relationship between F and $\widetilde{\mathcal{W}}_F$ in Sec. VIII of the appendix. The following theorem shows that $\widetilde{\mathcal{W}}_F := \{\widetilde{\Psi}_j, \widetilde{\Phi}_J\}_{j=0}^{J-1}$ is a nonexpansive operator under the assumption that the rows of $\mathbf{F}$ have disjoint support. Since softmax only leads to approximately sparse rows, this condition will in general only hold approximately. However, it can be easily achieved via post-processing, e.g. via hard sampling using the Gumbel-softmax [29] in each row of $\mathbf{F}$ as long as $J < t_{max}$.

Theorem 3: Suppose that the rows of $\mathbf{F}$ have disjoint support and that every element of $\text{supp}(\mathbf{F}_{j,\cdot})$ is less than every element of $\text{supp}(\mathbf{F}_{j+1,\cdot})$ for all $1 \leq j \leq J-1$. Then $\widetilde{\mathcal{W}}_F$ is a nonexpansive operator, i.e.

$$\sum_{j=0}^{J} \|\widetilde{\Psi}_j \mathbf{x}\|^2_{D^{-\frac{1}{2}}} + \|\widetilde{\Phi}_J \mathbf{x}\|^2_{D^{-\frac{1}{2}}} \leq \|\mathbf{x}\|^2_{D^{-\frac{1}{2}}} \tag{6}$$

The proof of Theorem 3 relies on applying Jensen's inequality and then Theorem 1. We provide full details in the Section IX of the appendix. We note that since the proof of Theorem 3 relies upon Jensen's inequality, we are not able to provide a lower bound in equation 6. However, we view the upper bound as more important for analyzing the theoretical properties of scattering transforms built using the filter bank $\widetilde{\mathcal{W}}_F$. Indeed, given Theorem 3, one may imitate the proof of Theorem 3.2 of [20] (and also analogous results in e.g., [11] and [14]) to easily derive upper bounds for scattering transforms constructed from $\widetilde{\mathcal{W}}_F$ (using the fact that $||a| - |b|| \leq |a - b|$). However, even if one were to obtain a lower-bound in equation 6, one would not be able to transfer this lower-bound to the scattering transform since $|\cdot|$ is not injective.

The aggregation submodule. While many approaches may be applied to aggregate node-level features into graph-level features such as max, mean, sum pooling, or the more powerful TopK [21] and attention pooling [30], we follow the statistical-moment aggregation explained in Secs. III-IV and leave exploration of other pooling methods to future work.These moments are a natural choice because they were shown to be effective in the (fixed) geometric scattering transform in [12]. The use of multiple moments together can be thought of as capturing the mean, standard deviation, and skew, and kurtosis of the $U_p x[v_i]$ over the graph (since these statistical quantities can all be computed from the first four moments of a random variable).

A. Incorporating LEGS Into a Larger Neural Network

As shown in [12] 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 two fully connected layers after the learnable scattering layers, which we denote LEGS-FCN, or using a modified RBF network [31], which we denote LEGS-RBF, to produce the final classification.

The latter configuration more accurately processes scattering features as shown in Table II. 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 $\text{RBF}(x) = \phi(\|\text{BatchNorm}(x) - c\|)$ with $\phi(x) = e^{-\|x\|^2}$. Where c is the SVM margin, which is set to 1 by default.

B. Backpropagation Through the LEGS Module

The LEGS module is fully suitable for incorporation in any neural network architecture and can be backpropagated through. To show this we write the partial derivatives with respect to the LEGS module parameters Θ and W here. The gradients of the filter $\widetilde{\Psi}_j, j \geq 1$, with respect to scale weights θ_k, $k = 1, \ldots J$, where σ is the softmax function, can be written as

$$\frac{\partial \widetilde{\Psi}_j}{\partial \Theta_{k,t}} = \begin{cases} P^t \sigma(\theta_j)_t (1 - \sigma(\theta_j))_t & \text{if } k = j, \\ P^t \sigma(\theta_j)_t \sigma(\theta_k)_t & \text{if } k = j+1, \\ 0_{n \times n} & \text{else.} \end{cases} \tag{7}$$

Finally, we note that while in this paper we consider the setting where W is fixed, one could also consider variations in which one attempted to learn W. In this case, the gradient matrix with respect to the adjacency matrix entry $W_{a,b}$ would be given by

$$\frac{\partial \widetilde{\Psi}_j}{\partial W_{a,b}} = \sum_{t=1}^{t_{\max}} \Bigg[(F_{j,t} - F_{j+1,t}) \times \frac{1}{2} \sum_{k=1}^{t} P^{k-1} \left(J^{ab} D^{-1} + W \frac{\partial D^{-1}}{\partial W_{a,b}} \right) P^{t-k} \Bigg], \tag{8}$$

where we denote J^{ab} the matrix with $J^{ab}_{a,b} = 1$ and all other entries zero, and $\frac{\partial D^{-1}}{\partial W_{a,b}}$ is defined in equation 17. In this setting, some tricks and heuristics may be needed to maintain desirable properties of W, (positivity, symmetry, etc.). Gradients of filters $\widetilde{\Phi}_J$ and $\widetilde{\Psi}_0$, which are simple modifications of the partial derivatives of $\widetilde{\Psi}_j$ and derivations of these gradients can be found in Sec. XI of the appendix.

VI. Empirical Results

We investigate the LEGS module on whole graph classification and graph regression tasks that arise in a variety of contexts, with emphasis on the more complex biochemical datasets. Unlike other types of data, biochemical graphs do not exhibit the small-world structure of social graphs and may have large diameters relative to 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 I
Dataset Statistics, Diameter, Nodes, Edges, Clustering Coefficient (CC) Averaged Over All Graphs. Split Into Bio-Chemical and Social Network Types

	# Graphs	# Classes	Diameter	Nodes	Edges	CC
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-B	1000	2	1.86	19.77	96.53	0.95
IMDB-M	1500	3	1.47	13.00	65.94	0.97
REDDIT-B	2000	2	8.59	429.63	497.75	0.05
REDDIT-12K	11929	11	9.53	391.41	456.89	0.03
REDDIT-5K	4999	5	10.57	508.52	594.87	0.03

A. Whole Graph Classification

We perform whole graph classification by using eccentricity (max distance of a node to other nodes) and clustering coefficient (percentage of links between the neighbors of the node compared to a clique) as node features as are used in [12]. We compare against graph convolutional networks (GCN) [4], GraphSAGE [5], graph attention network (GAT) [30], graph isomorphism network (GIN) [6], Snowball network [26], and fixed geometric scattering with a support vector machine classifier (GS-SVM) as in [12], 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 biochemical datasets and 6 social network datasets 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 I). For more specific information on individual datasets see Appendix XII. We use 10-fold cross validation on all models which is elaborated on in Section XIII of the supplement.

LEGS outperforms on biochemical datasets. Most work on graph neural networks has focused on social networks which have a well-studied structure. However, biochemical graphs that represent molecules and tend to be overall smaller and less connected than social networks (see Table I). In particular, we find that LEGS 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 II. On extremely small graphs we find that GS-SVM performs best, which is likely because other methods with more parameters can more easily overfit the data. We reason that the performance increase exhibited by LEGS module networks, and to a lesser extent GS-SVM, on these biochemical graphs is due the ability of geometric scattering to compute complex connectivity features via its multiscale diffusion wavelets. Thus,

TABLE II
ACCURACY (MEAN ± STD.) OVER 10 TEST SETS ON BIOCHEMICAL (TOP) AND SOCIAL NETWORK (BOTTOM) DATASETS

	LEGS-RBF	LEGS-FCN	LEGS-ATTN-FCN	LEGS-FIXED	GCN	GraphSAGE	GAT	GIN	GS-SVM	Baseline
DD	72.58 ± 3.35	72.07 ± 2.37	70.93 ± 4.81	69.09 ± 4.82	67.82 ± 3.81	66.37 ± 4.45	68.50 ± 3.62	42.37 ± 4.32	72.66 ± 4.94	**75.98 ± 2.81**
ENZYMES	36.33 ± 4.50	**38.50 ± 8.18**	33.72 ± 6.45	32.33 ± 5.04	31.33 ± 6.89	15.83 ± 9.10	25.83 ± 4.73	36.83 ± 4.81	27.33 ± 5.10	20.50 ± 5.99
MUTAG	33.51 ± 4.34	82.98 ± 9.85	84.60 ± 6.13	81.84 ± 11.24	79.30 ± 9.66	81.43 ± 11.64	79.85 ± 9.44	83.57 ± 9.68	**85.09 ± 7.44**	79.80 ± 9.92
NCI1	**74.26 ± 1.53**	70.83 ± 2.65	68.62 ± 2.37	71.24 ± 1.63	60.80 ± 4.26	57.54 ± 3.33	62.19 ± 2.18	66.67 ± 2.90	69.68 ± 2.38	56.69 ± 3.07
NCI109	**72.47 ± 2.11**	70.17 ± 1.46	64.58 ± 5.26	69.25 ± 1.75	61.30 ± 2.99	55.15 ± 2.58	61.28 ± 2.24	65.23 ± 1.82	68.55 ± 2.06	57.38 ± 2.20
PROTEINS	70.89 ± 3.91	71.06 ± 3.17	67.69 ± 4.31	67.30 ± 2.94	74.03 ± 3.20	71.87 ± 3.50	73.22 ± 3.55	**75.02 ± 4.55**	70.98 ± 2.67	73.22 ± 3.76
PTC	**57.26 ± 5.54**	56.92 ± 9.36	49.85 ± 11.37	54.31 ± 6.92	56.34 ± 10.29	55.22 ± 9.13	55.50 ± 6.90	55.82 ± 8.07	56.96 ± 7.09	56.71 ± 5.54
COLLAB	75.78 ± 1.95	75.40 ± 1.80	64.29 ± 5.82	72.94 ± 1.70	73.80 ± 1.73	**76.12 ± 1.58**	72.88 ± 2.06	62.98 ± 3.92	74.54 ± 2.32	64.76 ± 2.63
IMDB-BINARY	64.90 ± 3.48	64.50 ± 3.50	53.17 ± 4.35	64.30 ± 3.68	47.40 ± 6.24	46.40 ± 4.03	45.50 ± 3.14	64.20 ± 5.77	**66.70 ± 3.53**	47.20 ± 5.67
IMDB-MULTI	41.93 ± 3.01	40.13 ± 2.77	32.60 ± 9.52	41.67 ± 3.19	39.33 ± 3.13	39.73 ± 3.45	39.73 ± 3.61	38.67 ± 3.93	**42.13 ± 2.53**	39.53 ± 3.63
REDDIT-BINARY	**86.10 ± 2.92**	78.15 ± 5.42	81.74 ± 6.62	85.00 ± 1.93	81.60 ± 2.32	73.40 ± 4.38	73.35 ± 2.27	71.40 ± 6.98	85.15 ± 2.78	69.30 ± 5.08
REDDIT-MULTI-12K	38.47 ± 1.07	38.46 ± 1.31	28.26 ± 3.15	39.74 ± 1.31	**42.57 ± 0.90**	32.17 ± 2.04	32.74 ± 0.75	24.45 ± 5.52	39.79 ± 1.11	22.07 ± 0.98
REDDIT-MULTI-5K	47.83 ± 2.61	46.97 ± 3.06	41.37 ± 5.34	47.17 ± 2.93	**52.79 ± 2.11**	45.71 ± 2.88	44.03 ± 2.57	35.73 ± 8.35	48.79 ± 2.95	36.41 ± 1.80

methods that rely on a scattering construction would in general perform better, with the flexibility and trainability of the LEGS module giving it an edge on most tasks.

LEGS performs well on social network datasets and considerably improves performance in ensemble models. In Table II, we see that on the social network datasets LEGS performs well. We note that one of the scattering style networks (either LEGS or GS-SVM) is either the best or second best on each dataset. On each of these datasets, LEGS-RBF (which uses a SVM with a radial basis function similar to GS-SVM) and GS-SVM are well within one standard deviation of each other. If we also consider combining LEGS module features with GCN features the LEGS module performs the best on five out of six of the social network datasets. Across all datasets, an ensemble model considerably increases accuracy over GCN (see Table VIII). This underlines the capabilities of the LEGS module, not only as an isolated model, but also as a tool for powerful hybrid GNN architectures. Similar to [27], this supports the claim that the LEGS module (due to geometric scattering) is sensitive to complementary regularity over graphs, compared to many traditional GNNs. That is, many traditional GNNs focus on low-frequency information, whereas the band-pass wavelet filters are also able to capture high-frequency information. Since the two network types capture different information, using them together achieves better performance than either network by itself.

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 [12] where it was shown that the embeddings created through fixed geometric scattering can be used to accurately infer inter-graph relationships. Scattering features of enzyme graphs within the ENZYMES dataset [33] possessed sufficient global information to recreate the enzyme class exchange preferences observed empirically in [32]. We demonstrate here that LEGS features retain similar descriptive capabilities, as shown in Fig. 2. Here we show chord diagrams where the chord size represents the exchange preference between enzyme classes, which is estimated as suggested in [12]. Our results here (and in Table X, which provides complementary quantitative comparison) show that, with relaxations on the scattering parameters, LEGS-FCN achieves better

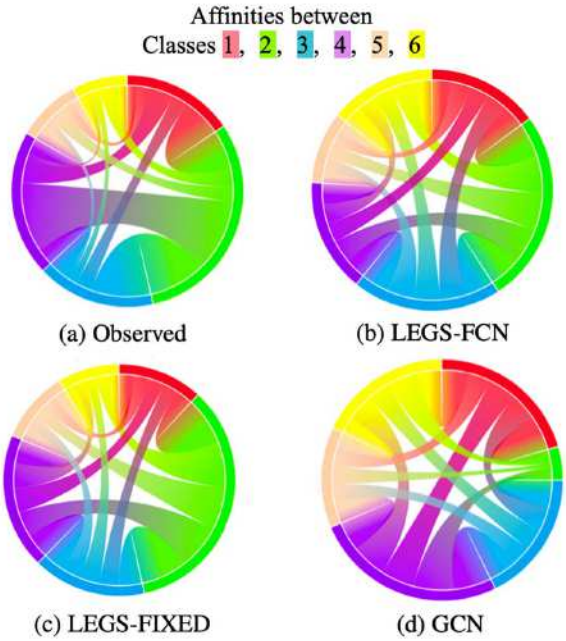


Fig. 2. Enzyme class exchange preferences empirically observed in [32], and estimated from LEGS and GCN embeddings.

classification accuracy than both LEGS-FIXED and GCN (see Table II) while also retaining a more descriptive embedding that maintains the global structure of relations between enzyme classes. We ran LEGS-FIXED and LEGS-FCN on the ENZYMES dataset. 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 [12] in defining the distance from an enzyme e to the enzyme class EC_j as $\mathrm{dist}(e, \mathrm{EC}_j) := \|v_e - \mathrm{proj}_{C_j}(v_e)\|$, where v_e 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, $\mathrm{mean}\{\mathrm{dist}(e, \mathrm{EC}_j) : e \in \mathrm{EC}_i\}$. From here, the affinity between two enzyme classes is computed as $\mathrm{pref}(\mathrm{EC}_i, \mathrm{EC}_j) = w_i / \min(\frac{D_{i,i}}{D_{i,j}}, \frac{D_{j,j}}{D_{j,i}})$, 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 [12] for handcrafted scattering,

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

TABLE IV
CASP GDT REGRESSION ERROR 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

LEGS-based networks are 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 III).

Ensembling LEGS with GCN features improves classification. Recent work by [27] combines the features from a fixed scattering transform with a GCN network, showing that this has empirical advantages in semi-supervised node classification, and theoretical representation advantages over a standard [4] style GCN. We ensemble the learned features from a learnable scattering network (LEGS-FCN) with those of GCN and compare this to ensembling fixed scattering features with GCN as in [27], as well as the solo features. Our setting is slightly different in that we use the GCN features from pretrained networks, only training a small 2-layer ensembling network on the combined graph level features. This network consists of a batch norm layer, a 128 width fully connected layer, a leakyReLU activation, and a final classification layer down to the number of classes. In Table VIII we see that combining GCN features with fixed scattering features in LEGS-FIXED or learned scattering features in LEGS-FCN always helps classification. Learnable scattering features help more than fixed scattering features overall and particularly in the biochemical domain.

B. Graph Regression

We next evaluate learnable scattering on two graph regression tasks, the QM9 [34] graph regression dataset, and a new task from the critical assessment of structure prediction (CASP) challenge [35]. In the CASP task, the main objective 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 [36]. 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 quantify their quality by the GDT score.

For this task we use the CASP12 dataset [35] and preprocess it similarly to [37], creating a KNN graph between proteins based on 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 interested in structure similarity and use no non-structural node features.

LEGS outperforms on all CASP targets. Across all CASP targets we find that LEGS-based architectures significantly outperform GNN and baseline models. This performance improvement is particularly stark on the easiest structures (measured by average GDT) but is consistent across all structures. In Fig. 3 we show the relationship between percent improvement of LEGS over the GCN model and the average GDT score across the target structures.

LEGS outperforms on the QM9 dataset. We evaluate the performance of LEGS-based networks on the quantum chemistry dataset QM9 [34], which consists of 130,000 molecules with ∼18 nodes per molecule. We use the node features from [34], with the addition of eccentricity and clustering coefficient features, and ignore the edge features. We whiten all targets to have zero mean and unit standard deviation. We train each network against all 19 targets as provided in the PyTorch geometric package [38], which includes the targets from [34] and [39]. and evaluate the mean squared error on the test set with mean and std. over four runs finding that LEGS improves over existing

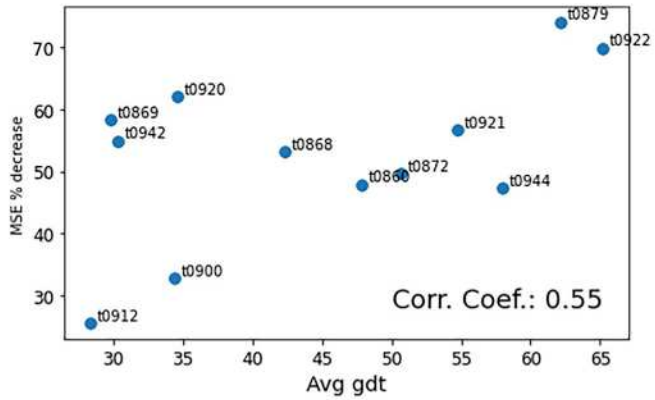


Fig. 3. CASP dataset LEGS-FCN % decrease over GCN in MSE of GDT prediction vs. average GDT score.

TABLE V
MEAN ± STD. OVER FOUR RUNS OF MEAN SQUARED ERROR OVER 19 TARGETS FOR THE QM9 DATASET, LOWER IS BETTER

$(\mu \pm \sigma)$	Test MSE
LEGS-FCN	**0.216 ± 0.009**
LEGS-ATTN-FCN	0.348 ± 0.106
LEGS-FIXED	0.228 ± 0.019
GraphSAGE	0.524 ± 0.224
GCN	0.417 ± 0.061
GIN	0.247 ± 0.037
Baseline	0.533 ± 0.041

models (Table V). On more difficult targets (i.e., those with large test error) LEGS-FCN is able to perform better, where on easy targets GIN is the best. We suspect this is due to difficult proteins having more long range connections. We also investigate performance on individual regression targets in Table XI.

Overall, scattering features offer a robust signal over many targets, and while perhaps less flexible (by construction), they achieve good average performance with significantly fewer parameters.

LEGS outperforms on the ZINC15 dataset. We compared the performance of LEGS-based networks against various architectures using 3 tranches of the ZINC15 [40] dataset in a multi-property prediction task (Table VI). The BBAB, FBAB and JBCD tranches contain molecules with molecular weight in the range of 200-250, 350-375 and 450-500 Daltons respectively. All networks were trained using the PyTorch geometric package [38], using a 1-hot encoding of atom type as the node signal. We predicted 10 chemical, physical and structural properties for each molecule, including measures of size, shape, lipophilicity, hydrogen bonding, and polarity. To illustrate the flexibility of the LEGS module, we also include a variant, LEGS-ATTN-FCN, which includes an attention layer between geometric scattering and the fully connected regression network and MP-LEGS-FCN, which applies LEGS to the output of a 3-layer graph messaging passing network. We observe that LEGS-ATTN-FCN and LEGS-FCN are the best performing methods

TABLE VI
MEAN ± STD. OVER FOUR RUNS OF MEAN SQUARED ERROR OVER 10 PROPERTIES IN ZINC, LOWER IS BETTER

$(\mu \pm \sigma)$	BBAB	FBAB	JBCD
LEGS-FCN	0.591 ± 0.026	**0.472 ± 0.018**	0.603 ± 0.022
LEGS-ATTN-FCN	**0.551 ± 0.033**	0.510 ± 0.029	**0.598 ± 0.072**
LEGS-FIXED	0.548 ± 0.017	0.496 ± 0.015	0.685 ± 0.017
MP-LEGS-FCN	1.033 ± 0.081	0.795 ± 0.024	0.802 ± 0.093
GraphSAGE	1.004 ± 0.037	0.994 ± 0.026	0.987 ± 0.011
GCN	0.841 ± 0.025	0.923 ± 0.037	0.892 ± 0.039
GIN	0.670 ± 0.018	0.673 ± 0.024	0.689 ± 0.022
Baseline	1.232 ± 0.146	1.399 ± 0.163	1.357 ± 0.154

TABLE VII
MEAN ± STD. OVER FOUR RUNS OF MEAN SQUARED ERROR IN BINDING AFFINITY PREDICTION OF TARGETS IN BINDINGDB, LOWER IS BETTER

$(\mu \pm \sigma)$	P00918	P14416
LEGS-FCN	**0.0318 ± 0.0009**	0.0441 ± 0.0013
LEGS-ATTN-FCN	0.0332 ± 0.0016	**0.0424 ± 0.0012**
LEGS-FIXED	0.0597 ± 0.0017	0.0514 ± 0.0020
MP-LEGS-FCN	0.1991 ± 0.0014	0.1429 ± 0.0034
GraphSAGE	0.1083 ± 0.0074	0.1106 ± 0.0087
GCN	0.1072 ± 0.0053	0.0994 ± 0.0039
GIN	0.0615 ± 0.0022	0.0583 ± 0.0036
Baseline	0.1137 ± 0.0076	0.1201 ± 0.0068

TABLE VIII
MEAN ± STD. TEST SET ACCURACY ON BIOCHEMICAL AND SOCIAL NETWORK DATASETS

$(\mu \pm \sigma)$	GCN	GCN-LEGS-FIXED	GCN-LEGS-FCN
DD	67.82 ± 3.81	**74.02 ± 2.79**	73.34 ± 3.57
ENZYMES	31.33 ± 6.89	31.83 ± 6.78	**35.83 ± 5.57**
MUTAG	79.30 ± 9.66	82.46 ± 7.88	**83.54 ± 9.39**
NCI1	60.80 ± 4.26	70.80 ± 2.27	**72.21 ± 2.32**
NCI109	61.30 ± 2.99	68.82 ± 1.80	**69.52 ± 1.99**
PROTEINS	74.03 ± 3.20	73.94 ± 3.88	**74.30 ± 3.41**
PTC	56.34 ± 10.29	**58.11 ± 6.06**	56.64 ± 7.34
COLLAB	73.80 ± 1.73	**76.60 ± 1.75**	75.76 ± 1.83
IMDB-BINARY	47.40 ± 6.24	65.10 ± 3.75	**65.90 ± 4.33**
IMDB-MULTI	39.33 ± 3.13	**39.93 ± 2.69**	39.87 ± 2.24
REDDIT-BINARY	81.60 ± 2.32	86.90 ± 1.90	**87.00 ± 2.36**
REDDIT-MULTI-12K	42.57 ± 0.90	45.41 ± 1.24	**45.55 ± 1.00**
REDDIT-MULTI-5K	52.79 ± 2.11	**53.87 ± 2.75**	53.41 ± 3.07

and achieve better performance compared to LEGS-FIXED, MP-LEGS-FCN, and other graph neural networks.

LEGS outperforms on the BindingDB dataset. We predicted the inhibition coefficient for ligands binding to 2 different targets in the BindingDB dataset [41], namely D(2) dopamine receptor (UniProtKB ID: P14416) and Carbonic anhydrase 2 (UniProtKB ID: P00918). The molecular weight and structural diversity of these ligands was significantly higher compared to the ZINC15 tranches. Learnable scattering networks (LEGS-FCN and LEGS-ATTN-FCN) outperformed GNN and baseline models (Tab. VII). Applying attention to the scattering output did not result in a significant performance improvement over the LEGS-FCN model, perhaps due to the FCN being capable of identifying scattering features pertaining to parts of ligand responsible for binding.

VII. Conclusion

Here we introduced a flexible geometric scattering module, that serves as an alternative to standard graph neural network architectures and is capable of learning rich multi-scale features. Our learnable geometric scattering module allows a task-dependent network to choose the appropriate scales of the multiscale graph diffusion wavelets that are part of the geometric scattering transform. We show that incorporation of this module yields improved performance on graph classification and regression tasks, particularly on biochemical datasets, while keeping strong guarantees on extracted features. This also opens the possibility to provide additional flexibility to the module to enable node-specific or graph-specific tuning via attention mechanisms, which are an exciting future direction, but out of scope for the current work.

Appendix

VIII. Illustration of Scale Selection Matrix

A traditional filter bank (Sec. III) $\mathcal{W}_J := \{\Psi_j, \Phi_J\}_{0\le j\le J}$, for the case $J=4$, would be the result of a scale selection matrix

$$\boldsymbol{F} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{J\times t_{\max}}.$$

However, as we derive $\boldsymbol{F}$ from a learned matrix $\boldsymbol{\Theta} \in \mathbb{R}^{J\times t_{\max}}$, we do not obtain an exact one-hot encoding per row. For example,

$$\boldsymbol{\Theta} \approx \begin{bmatrix} \mathbf{4.1} & 0.1 & 0.2 & 0.1 & 0.1 & 0.3 & 0.1 & 0.2 \\ 0.2 & 0.1 & \mathbf{5.2} & 0.1 & 0.1 & 0.3 & 0.1 & 0.2 \\ 0.1 & 0.1 & 0.3 & 0.1 & \mathbf{4.0} & 0.2 & 0.2 & 0.2 \\ 0.3 & 0.3 & 0.2 & 0.1 & 0.1 & 0.1 & 0.1 & \mathbf{5.3} \end{bmatrix}.$$

would yield $\boldsymbol{F} \approx$

$$\begin{bmatrix} \mathbf{0.88} & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.01 & 0.01 & \mathbf{0.96} & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.02 & 0.02 & 0.02 & 0.02 & \mathbf{0.87} & 0.02 & 0.02 & 0.02 \\ 0.0 & 0.011 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & \mathbf{0.96} \end{bmatrix}.$$

Then, the output wavelet $\widetilde{\Psi}_2$ of a learned filter bank (Sec. V) $\widetilde{\mathcal{W}}_{\boldsymbol{F}} := \{\widetilde{\Psi}_j, \widetilde{\Phi}_J\}_{j=0}^{J-1}$ for $J=4$ would have the form

$$\begin{aligned}\widetilde{\Psi}_2 \approx& -0.01\cdot \boldsymbol{P}^1 - 0.01\cdot \boldsymbol{P}^2 + \mathbf{0.94}\cdot \boldsymbol{P}^3 - 0.01\cdot \boldsymbol{P}^4 \\ &- \mathbf{0.86}\cdot \boldsymbol{P}^5 - 0.01\cdot \boldsymbol{P}^6 - 0.01\cdot \boldsymbol{P}^7 - 0.01\cdot \boldsymbol{P}^8 \\ \approx& \boldsymbol{P}^3 - \boldsymbol{P}^5.\end{aligned}$$

IX. The Proof of Theorem 3

Proof: Since each of the rows of $\mathbf{F}$ sums to one, we may define τ_j, $j=0,\dots,J$ to be an independent random variables with probability distribution given by $\mathbf{F}_{j,\cdot}$. Then, by definition

$$\widetilde{\Psi}_j \mathbf{x} = \sum_{t=1}^{t_{\max}} \mathbf{F}_{(j,t)}\mathbf{P}^t\mathbf{x} - \mathbf{F}_{(j+1,t)}\mathbf{P}^t\mathbf{x} = \mathbb{E}\left[\mathbf{P}^{\tau_j}\mathbf{x} - \mathbf{P}^{\tau_{j+1}}\mathbf{x}\right]$$

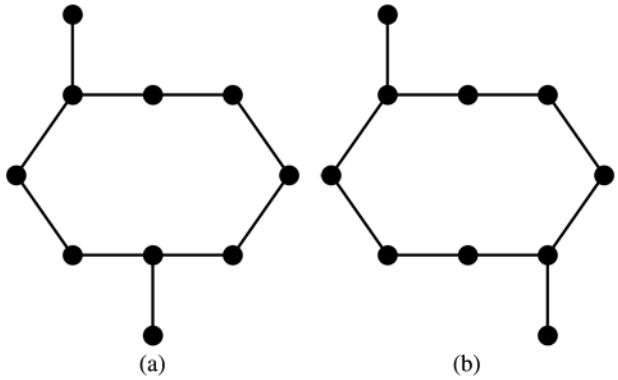


Fig. 4. Two graphs with similar structure consisting of an 8-cycle. Each cycle has two nodes that are connected to another node, which (with respected to shortest-path distance) are 3 steps (a) and 4 steps (b) apart, respectively.

for for all $1\le j\le J-1$. Similarly, we have

$$\widetilde{\Psi}_0\mathbf{x} = \mathbb{E}\left[\mathbf{x} - \mathbf{P}^{\tau_1}\mathbf{x}\right] \quad \text{and} \quad \widetilde{\Phi}_J\mathbf{x} = \mathbb{E}\mathbf{P}^{\tau_J}\mathbf{x}$$

Therefore, by Jensen's inequality we have

$$\begin{aligned}&\sum_{j=0}^{J} \|\widetilde{\Psi}_j\mathbf{x}\|^2_{\boldsymbol{D}^{-\frac{1}{2}}} + \|\widetilde{\Phi}_J\mathbf{x}\|^2_{\boldsymbol{D}^{-\frac{1}{2}}} && (9)\\ &= \|\mathbb{E}\left[\mathbf{x} - \mathbf{P}^{\tau_1}\mathbf{x}\right]\|^2_{\boldsymbol{D}^{-\frac{1}{2}}} && (10)\\ &\quad + \sum_{j=1}^{J-1} \|\mathbb{E}\left[\mathbf{P}^{\tau_j}\mathbf{x} - \mathbf{P}^{\tau_{j+1}}\mathbf{x}\right]\|^2_{\boldsymbol{D}^{-\frac{1}{2}}} + \|\mathbb{E}\mathbf{P}^{\tau_J}\mathbf{x}\|^2_{\boldsymbol{D}^{-\frac{1}{2}}} \\ &\le \mathbb{E}\left[\|\mathbf{x} - \mathbf{P}^{\tau_1}\mathbf{x}\|^2_{\boldsymbol{D}^{-\frac{1}{2}}} + \sum_{j=0}^{J} \|\mathbf{P}^{\tau_j}\mathbf{x} - \mathbf{P}^{\tau_{j+1}}\mathbf{x}\|^2_{\boldsymbol{D}^{-\frac{1}{2}}}\right. && (11)\\ &\quad \left. + \|\mathbf{P}^{\tau_J}\mathbf{x}\|^2_{\boldsymbol{D}^{-\frac{1}{2}}}\right]. && (12)\end{aligned}$$

By our assumptions on the support of the rows of $\mathbf{F}$, we have $\tau_j < \tau_{j+1}$ for all j. Therefore, the conditions of Theorem 1 are satisfied with probability one and so we have

$$\begin{aligned}&\|\mathbf{x} - \mathbf{P}^{\tau_1}\mathbf{x}\|^2_{\boldsymbol{D}^{-\frac{1}{2}}} + \sum_{j=0}^{J} \|\mathbf{P}^{\tau_j}\mathbf{x} - \mathbf{P}^{\tau_{j+1}}\mathbf{x}\|^2_{\boldsymbol{D}^{-\frac{1}{2}}} + \|\mathbf{P}^{\tau_J}\mathbf{x}\|^2_{\boldsymbol{D}^{-\frac{1}{2}}} \\ &\quad \le \|\mathbf{x}\|^2_{\boldsymbol{D}^{-\frac{1}{2}}}\end{aligned}$$

with probability one. Combining this with completes the proof. □

X. Case Study of Long-Range Dependencies

Here, we study the effect of the receptive field of different aggregations on the capacity to capture long-range interactions by the example of two graphs, shown in Fig. 4. We aim to distinguish two structurally similar graphs that both contain $n=12$ nodes and consist of an 8-cycle with each containing two nodes which are connected to an additional node. The major difference between the graphs is the shortest-path distance between those two nodes in terms of shortest-path distance, being 4 steps and 5 steps, respectively.

TABLE IX
TABLE INDICATING OF SIMPLE HANDCRAFTED FEATURE EXTRACTORS CAN DISTINGUISH THE TWO GRAPHS FROM FIG. 4 USING GCN AND SCATTERING AGGREGATIONS, RESPECTIVELY. ONLY SUFFICIENTLY LARGE RECEPTIVE FIELDS ALLOW TO DISTINGUISH THE GRAPHS

Recept. Field Radius R	Aggregation Matrices $\mathbf{M}$ $\mathbf{A}^R$ (GCN)	$P^1 - P^R$	$P^2 - P^R$	$P^3 - P^R$	$P^4 - P^R$
1	✗				
2	✗	✗			
3	✗	✓	✓		
4	✓	✓	✓	✓	
5	✓	✓	✓	✓	✓

To compare the capacity of different models to differentiate between the graphs, we create a simple feature extractor that leverages the aggregation scheme of a GCN model and of wavelets that resemble those learned by the LEGS module. In particular, we use the node degree $\mathbf{d} \in \mathbb{R}^n$ as input signal ($\mathbf{d}_i$ is the degree of node $v_i \in V$) and aggregate the signal according to the different approaches. Finally, we calculate the average of the output signal across the graph nodes to obtain a single value for each graph, i.e., $y = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{Md})_i$, with $\mathbf{M} \in \mathbb{R}^{n\times n}$ representing the different aggregation schemes.

For the GCN model, we use the matrix

$$\mathbf{A} := (\mathbf{D}+\mathbf{I})^{-1/2}(\mathbf{W}+\mathbf{I})(\mathbf{D}+\mathbf{I})^{-1/2}$$

as proposed in [4] and vary the radius $R > 0$ of the receptive field by setting $\mathbf{M} := \mathbf{A}^R$ for $R \in \mathbb{N}$. Similarly we experiment with scattering wavelets of different receptive fields that resemble the wavelets learned by the LEGS module by setting $\mathbf{M} := P^i - P^R$ with $1 \le i < R$.[1]

Table IX shows that the radius of the receptive field of the aggregation plays an important role in distinguishing the two graphs and that only methods with a receptive field of at least four are able to tell the graphs apart. Notably, GCN-style networks with large receptive fields tend not to perform well because of the oversmoothing problem, but wavelet-based methods do not suffer from this issue [27].

XI. GRADIENTS OF THE LEGS MODULE

Here we analyze the gradients of the LEGS module with respect to its outputs. As depicted in Fig. 1, the LEGS module has 3 inputs, W, Θ, and α, and J permutation equivariant output matrices $\widetilde{W}_{\boldsymbol{F}} = \{\widetilde{\boldsymbol{\Psi}}_j, \widetilde{\boldsymbol{\Phi}}_J\}_{j=0}^{J-1}$. Here we compute partial derivatives of $\widetilde{\boldsymbol{\Psi}}_j$ for $1 \le j \le J-1$. The other related gradients of $\widetilde{\boldsymbol{\Psi}}_0$ and $\widetilde{\boldsymbol{\Phi}}_J$ are easily deducible from these. The following partial derivatives and an application of the chain rule yield the gradients presented in the main paper:

$$\frac{\partial \widetilde{\boldsymbol{\Psi}}_j}{\partial \boldsymbol{F}_{k,t}} = \begin{cases} P^t & \text{if } k = j, \\ -P^t & \text{if } k = j+1, \\ 0_{n\times n} & \text{else,} \end{cases} \tag{13}$$

[1]The wavelets used for this analysis slightly differ from those learned by LEGs due to the construction of the scales illustrated in Section VIII. As a result, all learned wavelets effectively have a receptive field of radius $R = t_{\max}$, which further amplifies their ability to capture long-range dependencies.

$$\frac{\partial \widetilde{\boldsymbol{\Psi}}_j}{\partial P^t} = (\boldsymbol{F}_{j,t} - \boldsymbol{F}_{j+1,t}), \tag{14}$$

$$\frac{\partial \boldsymbol{F}_{k,t}}{\partial \boldsymbol{\Theta}_{k,t}} = \begin{cases} \sigma(\boldsymbol{\theta}_j)(1-\sigma(\boldsymbol{\theta}_j)) & \text{if } k = j, \\ -\sigma(\boldsymbol{\theta}_j)\sigma(\boldsymbol{\theta}_k) & \text{else,} \end{cases} \tag{15}$$

$$\frac{\partial P^t}{\partial \boldsymbol{W}_{ab}} = \frac{1}{2}\sum_{k=1}^{t} P^{k-1}\left(\boldsymbol{J}^{ab}\boldsymbol{D}^{-1} + \boldsymbol{W}\frac{\partial \boldsymbol{D}^{-1}}{\partial \boldsymbol{W}_{ab}}\right)P^{t-k}. \tag{16}$$

We note that in equation 16, we use the fact that $\boldsymbol{P} = \frac{1}{2}(\boldsymbol{I} + \boldsymbol{W}\boldsymbol{D}^{-1})$. The first term in the sum is constant, and to compute the derivative of the second term, we apply the product rule. The Jacobian of $\boldsymbol{W}$ with respect to coordinate ab is given by $\boldsymbol{J}^{ab}$ and the Jacobian of $\boldsymbol{D}^{-1}$ is given by

$$\left(\frac{\partial \boldsymbol{D}^{-1}}{\partial \boldsymbol{W}_{ab}}\right)_{ij} = \begin{cases} -\left(\sum_j \boldsymbol{W}_{aj}\right)^{-2} & \text{if } i = j = a \\ 0 & \text{else} \end{cases} \tag{17}$$

XII. DATASETS

In this section we provide further information and analysis on the individual datasets which relates the composition of the dataset as shown in Table I to the relative performance of our models as shown in Table II. Datasets used in Tables VI and VII are also described here.

DD [42] is a dataset extracted from the protein data bank (PDB) of 1178 high resolution proteins where each node represents an amino acid and nodes are connected if they are proximal in 3D space or along the linear protein backbone. 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 [33] is a dataset of 600 enzymes divided into 6 balanced classes of 100 enzymes each. Each node represents an amino acid and edges are present either in case of linkage along the protein backbone or if two amino acids are within close proximity. 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 [43] contain slight variants of 4100 chemical compounds encoded as graphs where each node represents an atom and edges represent bonds. 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 have 30 nodes with a similar number of edges. This makes for long graphs with high diameter.

PROTEINS [33] contains 1178 protein structures with the goal of classifying enzymes vs. non enzymes. Nodes represent secondary structure elements and two nodes are connected if they are adjacent on the backbone or less than six Angstroms apart. 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 [44] contains 344 chemical compound graphs divided into two classes based on whether or not they cause cancer in rats. Here each node is an atom and each edge is an atomic bond. 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 [45] contains 5000 ego-networks of different researchers from high energy physics, condensed matter physics or astrophysics. Here each node is a researcher and edges represent co-authorship. 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 [45] contains graphs with nodes representing actresses/actors and edges 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 [45] consists of three independent datasets. In REDDIT-BINARY/MULTI-5K/MULTI-12K, 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.

ZINC15 [40] is a database of small drug-like molecules for virtual screening. Here each node represents an atom and each edge represents a bond between two atoms. The data is organized into 2D tranches consisting of approximately 997 million molecules categorized by molecular weight, solubility (LogP), reactivity and availability for purchase.

BindingDB [41] is a publicly available database of binding affinities, focusing on interactions between small, drug-like ligands and proteins considered to be candidate drug-targets. BindingDB contains approximately 2.5 million interactions between more than 8,000 proteins and 1 million drug-like molecules.

QM9 [34], [39] is a dataset of stable organic molecules with up to 9 heavy atoms. Each node is a heavy atom with edges representing bonds between heavy atoms.

CASP We use the CASP-12 dataset where each graph represents a protein with nodes connected if they are close in 3D space or connected along the backbone. We include the amino acid type as a node feature.

XIII. Training Details

We train all models for a maximum of 1000 epochs with an initial learning rate of 10^{-4} using the ADAM optimizer [46]. We terminate training if validation loss does not improve for 100 epochs testing every 10 epochs. Our models are implemented with Pytorch [47] and Pytorch geometric [38]. 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 [12]. Most experiments were run on a 2×18 core Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz server with 512GB of RAM equipped with two Nvidia TITAN RTX gpus.

A. 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.

TABLE X
Quantified Distance Between the Empirically Observed Enzyme Class Exchange Preferences of [32]

LEGS-FIXED	LEGS-FCN	GCN
0.132	0.146	0.155

XIV. Additional Experiments

Quantification of the enzyme class exchange preferences We quantify the empirically observed enzyme class exchange preferences of [32] and the class exchange preferences inferred from LEGS-FIXED, LEGS-FCN, and a GCN in Table X. We measure the cosine distance between the graphs represented by the chord diagrams in Fig. 2. As before, the self-affinities were discarded. We observe that LEGS-Fixed is best able to reproduces the exchange preferences. However, LEGS-FCN still reproduces the observed exchange preferences well and has significantly better classification accuracy than LEGS-Fixed.

QM9 Target Breakdown [34], [39] contains graphs that each represent chemicals with $\tilde{1}8$ atoms. Regression targets represent chemical properties of the molecules. These targets are respectively, the dipole moment μ, the isotropic polarizability α, the highest occupied molecular orbital energy ϵ_{HOMO}, the lowest unoccupied molecular orbital energy ϵ_{LUMO}, the difference $\Delta\epsilon = \epsilon_{\text{HOMO}} - \epsilon_{\text{LUMO}}$, the electronic spatial extent, $\langle R^2 \rangle$, the zero point vibrational energy ZPVE, the internal energy at 0K U_0, the internal energy U, the enthalpy H, the free energy G, and the heat capacity c_{v} at 25C, the atomization energy at 0K U_0^{ATOM} and 25C U^{ATOM}, the atomization enthalpy H^{ATOM} and free energy G^{ATOM} at 25C, and three rotational constants

TABLE XI
MEAN ± STD. OVER FOUR RUNS OF MEAN SQUARED ERROR OVER 19 TARGETS FOR THE QM9 DATASET, LOWER IS BETTER

	LEGS-FCN	LEGS-FIXED	GCN	GraphSAGE	GIN	Baseline
μ	**0.749 ± 0.025**	0.761 ± 0.026	0.776 ± 0.021	0.876 ± 0.083	0.786 ± 0.032	0.985 ± 0.020
α	**0.158 ± 0.014**	0.164 ± 0.024	0.448 ± 0.007	0.555 ± 0.295	0.191 ± 0.060	0.593 ± 0.013
ϵ_{HOMO}	**0.830 ± 0.016**	0.856 ± 0.026	0.899 ± 0.051	0.961 ± 0.057	0.903 ± 0.033	0.982 ± 0.027
ϵ_{LUMO}	0.511 ± 0.012	**0.508 ± 0.005**	0.549 ± 0.010	0.688 ± 0.216	0.555 ± 0.006	0.805 ± 0.025
$\Delta\epsilon$	**0.587 ± 0.007**	**0.587 ± 0.006**	0.609 ± 0.009	0.755 ± 0.177	0.613 ± 0.013	0.792 ± 0.010
$\langle R^2 \rangle$	**0.646 ± 0.013**	0.674 ± 0.047	0.889 ± 0.014	0.882 ± 0.118	0.699 ± 0.033	0.833 ± 0.026
ZPVE	0.018 ± 0.012	0.020 ± 0.011	0.099 ± 0.011	0.321 ± 0.454	**0.012 ± 0.006**	0.468 ± 0.005
U_0	0.017 ± 0.005	0.024 ± 0.008	0.368 ± 0.015	0.532 ± 0.405	**0.015 ± 0.005**	0.379 ± 0.013
U	0.017 ± 0.005	0.024 ± 0.008	0.368 ± 0.015	0.532 ± 0.404	**0.015 ± 0.005**	0.378 ± 0.013
H	0.017 ± 0.005	0.024 ± 0.008	0.368 ± 0.015	0.532 ± 0.404	**0.015 ± 0.005**	0.378 ± 0.013
G	0.017 ± 0.005	0.024 ± 0.008	0.368 ± 0.015	0.533 ± 0.404	**0.015 ± 0.005**	0.380 ± 0.014
c_v	**0.254 ± 0.013**	0.279 ± 0.023	0.548 ± 0.023	0.617 ± 0.282	0.294 ± 0.003	0.631 ± 0.013
U_0^{ATOM}	0.034 ± 0.014	0.033 ± 0.010	0.215 ± 0.009	0.356 ± 0.437	**0.020 ± 0.002**	0.478 ± 0.014
U^{ATOM}	0.033 ± 0.014	0.033 ± 0.010	0.214 ± 0.009	0.356 ± 0.438	**0.020 ± 0.002**	0.478 ± 0.014
H^{ATOM}	0.033 ± 0.014	0.033 ± 0.010	0.213 ± 0.009	0.355 ± 0.438	**0.020 ± 0.002**	0.478 ± 0.014
G^{ATOM}	0.036 ± 0.014	0.036 ± 0.011	0.219 ± 0.009	0.359 ± 0.436	**0.023 ± 0.002**	0.479 ± 0.014
A	0.002 ± 0.002	0.001 ± 0.001	0.017 ± 0.034	0.012 ± 0.022	**0.000 ± 0.000**	0.033 ± 0.013
B	0.083 ± 0.047	**0.079 ± 0.033**	0.280 ± 0.354	0.264 ± 0.347	0.169 ± 0.206	0.205 ± 0.220
C	**0.062 ± 0.005**	0.176 ± 0.231	0.482 ± 0.753	0.470 ± 0.740	0.321 ± 0.507	0.368 ± 0.525

A, B, C measured in gigahertz. For more information see references [34], [38], [39]. In Table XI we split out performance by target. GIN performs slightly better on the molecule energy targets both overall and atomization targets U_0, U, H, and G where both LEGS and GIN significantly outperform the other models GCN, GraphSAGE and the structure invariant baseline. On all other targets, and especially the more difficult targets (measured by baseline MSE) the LEGS module performs the best or near to the best.

ACKNOWLEDGMENT

The content provided here is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies.

REFERENCES

[1] M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.
[2] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. Int. Conf. Learn. Representations*, 2014.
[3] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst. 29*, 2016.
[4] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2016.
[5] W. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst. 30*, 2017.
[6] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. Int. Conf. Learn. Representations*, 2019.
[7] S. Abu-El-Haija et al., "MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 21–29.
[8] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. AAAI Conf. Artif. Intell.*, 2018.
[9] P. Barceló, E. Kostylev, M. Monet, J. Pérez, J. Reutter, and J. Silva, "The logical expressiveness of graph neural networks," in *Proc. Int. Conf. Learn. Representations*, 2020.
[10] U. Alon and E. Yahav, "On the bottleneck of graph neural networks and its practical implications," in *Proc. Int. Conf. Learn. Representations*, 2021.
[11] S. Mallat, "Group invariant scattering," *Commun. Pure Appl. Math.*, vol. 65, no. 10, pp. 1331–1398, Oct. 2012.
[12] F. Gao, G. Wolf, and M. Hirn, "Geometric scattering for graph data analysis," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 2122–2131.
[13] F. Gama, A. Ribeiro, and J. Bruna, "Diffusion scattering transforms on graphs," in *Proc. Int. Conf. Representation Learn.*, 2019.
[14] D. Zou and G. Lerman, "Graph convolutional neural networks via scattering," *Appl. Comput. Harmon. Anal.*, vol. 49, no. 3, pp. 1046–1074, Nov. 2019.
[15] M. Perlmutter, F. Gao, G. Wolf, and M. Hirn, "Geometric wavelet scattering networks on compact riemannian manifolds," in *Proc. Math. Sci. Mach. Learn.*, PMLR, 2020, pp. 570–604.
[16] J. D. McEwen, C. G. Wallis, and A. N. Mavor-Parker, "Scattering networks on the sphere for scalable and rotationally equivariant spherical CNNs," 2021, *arXiv:2102.02828*.
[17] J. Chew et al., "The manifold scattering transform for high-dimensional point cloud data," in *Proc. Topol., Algebr., Geometric Learn. Workshops*, PMLR, 2022, pp. 67–78.
[18] R. Coifman and M. Maggioni, "Diffusion wavelets," *Appl. Comput. Harmon. Anal.*, vol. 21, no. 1, pp. 53–94, 2006.
[19] F. Gama, J. Bruna, and A. Ribeiro, "Stability of graph scattering transforms," in *Proc. Adv. Neural Inf. Process. Syst. 32*, 2019.
[20] M. Perlmutter, A. Tong, F. Gao, G. Wolf, and M. Hirn, "Understanding graph neural networks with generalized geometric scattering transforms," 2023, *arXiv:1911.06253*.
[21] H. Gao and S. Ji, "Graph U-Nets," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, PMLR, 2019, pp. 2083–2092.
[22] A. Tong, F. Wenkel, K. MacDonald, S. Krishnaswamy, and G. Wolf, "Data-driven learning of geometric scattering networks," in *IEEE IEEE Int. Workshop Mach. Learn. Signal Process. (MLSP)*, 2021, pp. 1–6.
[23] M. Balcilar, G. Renton, P. Héroux, B. Gaüzere, S. Adam, and P. Honeine, "Analyzing the expressive power of graph neural networks in a spectral perspective," in *Proc. Int. Conf. Learn. Representations*, 2020.
[24] H. Nt and T. Maehara, "Revisiting graph neural networks: All we have is low-pass filters," 2019, *arXiv:1905.09550*.
[25] R. Liao, Z. Zhao, R. Urtasun, and R. S. Zemel, "LanczosNet: Multi-scale deep graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2019.
[26] S. Luan, M. Zhao, X. Chang, and D. Precup, "Break the ceiling: Stronger multi-scale deep graph convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst. 32*, 2019.
[27] F. Wenkel, Y. Min, M. Hirn, M. Perlmutter, and G. Wolf, "Overcoming oversmoothness in graph convolutional networks via hybrid scattering networks," 2022, *arXiv:2201.08932*.
[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. Accessed: Jan. 27, 2023. [Online]. Available: http://www.deeplearningbook.org
[29] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-Softmax," *Proc. Int. Conf. Learn. Representations*, 2016.
[30] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *Proc. Int. Conf. Representation Learn.*, 2018.
[31] D. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Syst.*, vol. 2, pp. 321–355, 1988.
[32] S. M. Cuesta, S. A. Rahman, N. Furnham, and J. M. Thornton, "The classification and evolution of enzyme function," *Biophys. J.*, vol. 109, no. 6, pp. 1082–1086, 2015.
[33] 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*, vol. 21, pp. i47–i56, 2005.
[34] J. Gilmer, S. Schoenholz, P. Riley, O. Vinyals, and G. Dahl, "Neural message passing for quantum chemistry," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, PMLR, 2017, pp. 1263–1272.
[35] J. Moult, K. Fidelis, A. Kryshtafovych, T. Schwede, and A. Tramontano, "Critical assessment of methods of protein structure prediction (CASP)—Round XII," *Proteins*, vol. 86, no. 1, pp. 7–15, Dec. 2018.
[36] V. Modi, Q. Xu, S. Adhikari, and R. Dunbrack, "Assessment of template-based modeling of protein structure in CASP11," *Proteins*, vol. 84, no. 1, pp. 200–220, Jun. 2016.
[37] J. Ingraham, V. Garg, R. Barzilay, and T. Jaakkola, "Generative models for graph-based protein design," in *Proc. Adv. Neural Inf. Process. Syst. 32*, 2019.
[38] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch geometric," in *Proc. ICLR Workshop Representation Learn. Graphs Manifolds*, 2019.
[39] Z. Wu et al., "MoleculeNet: A benchmark for molecular machine learning," *Chem. Sci.*, vol. 9, no. 2, pp. 513–530, Oct. 2018.

[40] J. J. Irwin and B. K. Shoichet, "ZINC—A free database of commercially available compounds for virtual screening," *J. Chem. Inf. Model.*, vol. 45, no. 1, pp. 177–182, 2005. Accessed: Jan. 27, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1360656/

[41] M. K. Gilson, T. Liu, M. Baitaluk, G. Nicola, L. Hwang, and J. Chong, "BindingDB in 2015: A public database for medicinal chemistry, computational chemistry and systems pharmacology," *Nucleic Acids Res.*, vol. 44, no. D1, pp. 1045–1053, Jan. 2016.

[42] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," *J. Mol. Biol.*, vol. 330, no. 4, pp. 771–783, 2003. Accessed: Jan. 27, 2023. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022283603006284

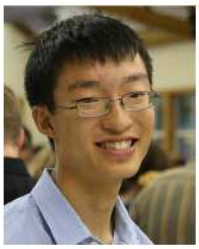
[43] N. Wale, I. A. Watson, and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," *Knowl. Inf. Syst.*, vol. 14, no. 1, p. 2007, Aug. 2008.

[44] H. Toivonen, A. Srinivasan, R. D. King, S. Kramer, and C. Helma, "Statistical evaluation of the predictive toxicology challenge 2000-2001," *Bioinformatics*, vol. 19, no. 10, pp. 1183–1193, 2003.

[45] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD '15)*, Sydney, NSW, Australia, 2015, pp. 1365–1374. Accessed: Jan. 27, 2023. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2783258.2783417

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, *arXiv:1412.6980*.

[47] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst. 33*, 2019, pp. 8026–8037.



Alexander Tong received the M.Phil. and Ph.D. degrees in computer science from Yale University. He is a Postdoctoral Fellow with the Department of Computer Science and Operations Research (DIRO), Université de Montréal and also with Mila (the Quebec AI institute). His research interests include machine learning and algorithms. His research focuses on optimal transport and neural network methods for geometric single-cell data using a combination of graph and deep learning methods.

Frederik Wenkel received the B.Sc. and M.Sc. degrees in mathematics from the Technical University of Munich, in 2019. He is currently working toward the Ph.D. degree in applied mathematics with the Université de Montréal (UdeM) and Mila (Quebec AI Institute), working on geometric deep learning. His research interests include graph neural networks and their applications in domains such as social networks and bio-chemistry.

Dhananjay Bhaskar (Member, IEEE) received the Ph.D. degree in biomedical engineering from Brown University, in 2021. He is a Postdoctoral Research Associate with the Department of Genetics, Yale University and a Yale-Boehringer Ingelheim Biomedical Data Science Fellow. His research interests include mathematical modeling, topological data analysis and machine learning for the analysis of multimodal and multiscale data in a variety of biological systems.

Kincaid MacDonald is an undergraduate with Yale studying mathematics, computer science, philosophy, and exploring their practical intersection in artificial intelligence. His research interests include developing mathematical tools to shine light into the "black box" of modern deep neural networks.

Jackson Grady is an undergraduate with Yale College, currently working toward the B.S. degree in computer science. He joined the Krishnaswamy Lab in Summer 2021 to explore his research interests in machine learning. His research interests include informative representations of graph-structured data and its applications to tackling problems such as drug discovery and analysis of protein folding.

Michael Perlmutter received the Ph.D. degree in mathematics from Purdue University, in 2016. He is an Assistant Professor with the Department of Mathematics, Boise State University. Previously, he has held Postdoctoral positions with the Department of Mathematics, UCLA, the Department of Computational Mathematics, Science and Engineering, Michigan State University, and the Department of Statistics and Operations Research, the University of North Carolina at Chapel Hill. His research interests include applied probability and computational harmonic analysis to develop an analyze new methods for data sets with geometric structure.

Smita Krishnaswamy (Member, IEEE) is an Associate Professor with the Department of Genetics, Yale School of Medicine and with the Department of Computer Science, Yale School of Applied Science and Engineering, and a Core Member of the Program in applied mathematics. She is also with Yale Center for Biomedical Data Science, Yale Cancer Center, and Program in Interdisciplinary Neuroscience. Her research interests include developing unsupervised machine learning methods (especially graph signal processing and deep-learning) to denoise, impute, visualize and extract structure, patterns and relationships from big, high throughput, high dimensional biomedical data. Her methods have been applied variety of datasets from many systems including embryoid body differentiation, zebrafish development, the epithelial-to-mesenchymal transition in breast cancer, lung cancer immunotherapy, infectious disease data, gut microbiome data and patient data. She was trained as a Computer Scientist with the Ph.D. degree from the University of Michigan's EECS Department where her research focused on algorithms for automated synthesis and probabilistic verification of nanoscale logic circuits. Following her time in Michigan, she spent 2 years with IBM's TJ Watson Research Center as a Researcher in the systems division where she worked on automated bug finding and error correction in logic.

Guy Wolf received the M.Sc. and Ph.D. degrees in computer science from Tel Aviv University. He is an Associate Professor with the Department of Mathematics and Statistics (DMS), Université de Montréal (UdeM), a Core Academic Member of Mila (the Quebec AI institute), and holds a Canada CIFAR AI Chair. He is also with the CRM Center of Mathematical Sciences and the IVADO Institute of data valorization. Prior to joining UdeM in 2018, he was a Postdoctoral Researcher with the Department of Computer Science, École Normale Supérieure in Paris (France) from 2013 to 2015, and a Gibbs Assistant Professor in the applied mathematics program with Yale University from 2015. His research interests include manifold learning and geometric deep learning for exploratory data analysis, including methods for dimensionality reduction, visualization, denoising, data augmentation, and coarse graining. Further, he is particularly interested in biomedical data exploration applications of such methods, e.g., in single cell genomics/proteomics and neuroscience.