DATA-DRIVEN LEARNING OF GEOMETRIC SCATTERING MODULES FOR GNNS

Alexander Tong[1*] *Frederick Wenkel*[2*] *Kincaid Macdonald*[3] *Smita Krishnaswamy*[4,1†] *Guy Wolf*[2†]

[1] Yale University, Dept. of Comp. Sci.; [3] Dept. of Math.; [4] Dept. of Genetics, New Haven, CT, USA
[2] Université de Montréal, Dept. of Math. & Stat.; Mila – Quebec AI Institute, Montreal, QC, Canada

2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP) | 978-1-7281-6338-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/MLSP52302.2021.9596169

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.

Index Terms— Geometric deep learning, graph neural networks, geometric scattering

1. 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] . However, recent work has shown the difficulty in generalizing these methods to more complex structures, identifying common problems and phrasing them in terms of oversmoothing [5], oversquashing [6] or underreaching [7].

Recently, an alternative approach was presented to provide deep geometric representation learning by generalizing Mallat's scattering transform [8], originally proposed to provide a mathematical framework for understanding convolutional neural networks, to graphs [9, 10, 11] and manifolds [12]. 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 as diffusion wavelets over the input graph [13], which are then cascaded with pointwise absolute-value nonlinearities. The efficacy of geometric scattering features in graph processing tasks was demonstrated in [9], 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 [14, 15].

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.

2. PRELIMINARIES: GEOMETRIC SCATTERING

Let $\mathcal{G} = (V, E, w)$ be a weighted graph with $V \coloneqq \{v_1, \dots, v_n\}$ the set of nodes, $E \subset \{\{v_i, v_j\} \in V \times V, i \neq j\}$ the set of (undirected) edges and $w : E \to (0, \infty)$ assigning (positive) edge weights to the graph edges. We define a *graph signal* as a function $x : V \to \mathbb{R}$ on the nodes of $\mathcal{G}$ and aggregate them in a signal vector $\boldsymbol{x} \in \mathbb{R}^n$ with the 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 $\boldsymbol{W}[v_i, v_j] \coloneqq w(v_i, v_j)$ 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 $\boldsymbol{D} \coloneqq \mathrm{diag}(d_1, \dots, d_n)$ with $d_i \coloneqq \deg(v_i) \coloneqq \sum_{j=1}^{n} \boldsymbol{W}[v_i, v_j]$ the *degree* of node v_i.

The geometric scattering transform [9] consists of a cas-

* Equal contribution. † Equal senior author contribution. This research was partially funded by IVADO grant PRF-2019-3583139727, FRQNT grant 299376, CIFAR AI Chair [*G.W.*]; CZI grants 182702 & CZF2019-002440 [*S.K.*]; and NIH grants R01GM135929 & R01GM130847 [*G.W., S.K.*]. The content provided here is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies. Correspondence to: smita.krishnaswamy@yale.edu and guy.wolf@umontreal.ca

cade of graph filters constructed from a left stochastic diffusion matrix $\boldsymbol{P} \coloneqq \frac{1}{2}\big(\boldsymbol{I}_n + \boldsymbol{W}\boldsymbol{D}^{-1}\big)$, which corresponds to transition probabilities of a lazy random walk Markov process. The laziness of the process signifies that at each step it has equal probability of 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

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

These diffusion wavelet operators partition the frequency spectrum into dyadic frequency bands, which are then organized into a full wavelet filter bank $\mathcal{W}_J \coloneqq \{\boldsymbol{\Psi}_j, \boldsymbol{\Phi}_J\}_{0\leq j\leq J}$, where $\boldsymbol{\Phi}_J \coloneqq \boldsymbol{P}^{2^J}$ is a pure low-pass filter, similar to the one used in GCNs. It is easy to verify that the resulting wavelet transform is invertible, since a simple sum of filter matrices in $\mathcal{W}_J$ yields the identity. Moreover, as discussed in [15], 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 [10, 14] and [11].

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

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

parameterized by the scattering path $p \coloneqq (j_1, \dots, j_m) \in \cup_{m\in\mathbb{N}}\mathbb{N}_0^m$ that determines the filter scales of each wavelet. Whole-graph representations are obtained by aggregating node-level features via statistical moments over the nodes of the graph [9], which yields the geometric scattering features

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

indexed by the scattering path p and moment order q. Finally, we note that it can be shown that the graph-level scattering transform $\boldsymbol{S}_{p,q}$ guarantees node-permutation invariance, while $\boldsymbol{U}_p$ is permutation equivariant [15, 9].

3. ADAPTIVE GEOM. SCATTERING RELAXATION

The geometric scattering construction, described in Sec. 2, 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 [15], while also providing effective unsupervised graph representation learning for data exploration, which also has advantages in supervised learning tasks [9]. While the handcrafted design in [15] and [9] is not a priori amenable to task-driven tuning provided by end-to-end GNN training, we note that the cascade in Eq. 3 does conform to a neural network architecture suitable for backpropagation. Therefore, in this section, we show how and under what conditions a relaxation of the selection of the scales preserves some of the useful mathematical properties established in [15].

We replace the handcrafted dyadic scales in Eq. 1 with an adaptive monotonic sequence of integer diffusion time scales $0 < t_1 < \dots < t_J$, which are selected via training. The adaptive filter bank $\mathcal{W}'_J \coloneqq \{\boldsymbol{\Psi}'_j, \boldsymbol{\Phi}'_J\}_{j=0}^{J-1}$, contains wavelets

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

The following theorem shows that for any selection of scales, the relaxed construction of $\mathcal{W}'_J$ yields a nonexpansive frame, similar to the result from [15] shown for the original handcrafted construction.

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 $\boldsymbol{M} \coloneqq \boldsymbol{D}^{-1/2}\boldsymbol{P}\boldsymbol{D}^{1/2}$ with eigendecomposition $\boldsymbol{M} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$ for orthogonal $\boldsymbol{Q}$. Given this decomposition, we can write

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

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

$$\|\boldsymbol{\Phi}'_J\boldsymbol{x}\|^2_{\boldsymbol{D}^{-1/2}} = \langle\boldsymbol{\Phi}'_J\boldsymbol{x}, \boldsymbol{\Phi}'_J\boldsymbol{x}\rangle_{\boldsymbol{D}^{-1/2}} = \|\boldsymbol{\Lambda}^{t_J}\boldsymbol{Q}^T\boldsymbol{D}^{-1/2}\boldsymbol{x}\|_2^2.$$

If we consider a change of variable to $\boldsymbol{y} = \boldsymbol{Q}^T\boldsymbol{D}^{-1/2}\boldsymbol{x}$, we have $\|\boldsymbol{x}\|^2_{\boldsymbol{D}^{-1/2}} = \|\boldsymbol{D}^{-1/2}\boldsymbol{x}\|_2^2 = \|\boldsymbol{y}\|_2^2$, while $\|\boldsymbol{\Phi}'_J\boldsymbol{x}\|^2_{\boldsymbol{D}^{-1/2}} = \|\Lambda^{t_J}\boldsymbol{y}\|_2^2$. Similarly, we can also reformulate the operations of the other filters in terms of diagonal matrices applied to $\boldsymbol{y}$ as $\|\boldsymbol{\Psi}'_j\boldsymbol{x}\|^2_{\boldsymbol{D}^{-1/2}} = \|(\Lambda^{t_j} - \Lambda^{t_{j+1}})\boldsymbol{y}\|_2^2$.

Given these reformulations, we can now write

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

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

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

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 $C \coloneqq \min_{0\leq\xi\leq 1}(\xi^{2t_J} + (1-\xi^{t_1})^2) > 0$, 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. □

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

The following theorem establishes that any such configuration, extracted from $\mathcal{W}'_J$ via Eq. 2-3, is permutation equivariant at the node-level and permutation invariant at the graph level. This guarantees that the extracted (in this case learned) features indeed encode intrinsic graph geometry rather than a priori indexation.

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

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

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

$$\overline{\boldsymbol{\Psi}}_j\Pi\boldsymbol{x} = (\Pi\boldsymbol{P}^{t_j}\Pi^T - \Pi\boldsymbol{P}^{t_{j+1}}\Pi^T)\Pi\boldsymbol{x} = \Pi\boldsymbol{\Psi}_j\boldsymbol{x},$$

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 [15]. We carefully account here for the relaxed learned configuration, which replaces the original handcrafted one there.

4. A LEARNABLE GEOM. SCATTERING MODULE

The adaptive geometric scattering construction presented in Sec. 3 is implemented here 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 $\boldsymbol{X} \in \mathbb{R}^{n\times N_{\ell-1}}$. 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 $\boldsymbol{x}$ by iteratively multiplying the diffusion matrix $\boldsymbol{P}$ to the left of the signal, resulting in $\left[\boldsymbol{P}\boldsymbol{x}, \boldsymbol{P}^2\boldsymbol{x}, \ldots, \boldsymbol{P}^m\boldsymbol{x}\right]$. Since $\boldsymbol{P}$ is often sparse, for efficiency reasons these filter responses are implemented via an RNN structure consisting of m RNN modules. Each module propagates the incoming hidden state $\boldsymbol{h}_{t-1}, t = 1, \ldots, m$ with $\boldsymbol{P}$ with the readout $\boldsymbol{o}_t$ equal to the produced hidden state, $\boldsymbol{h}_t \coloneqq \boldsymbol{P}\boldsymbol{h}_{t-1}, \quad \boldsymbol{o}_t \coloneqq \boldsymbol{h}_t$.

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. 2 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).

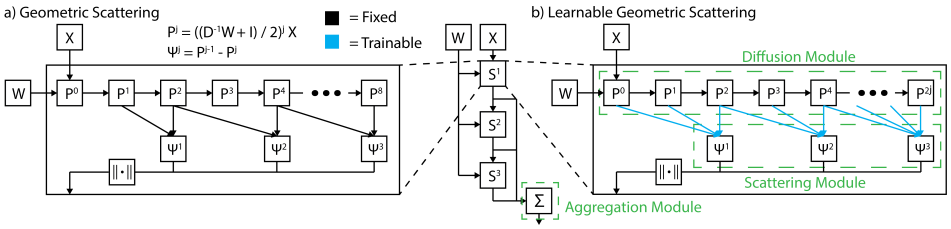

Fig. 1: LEGS module learns to select the appropriate scattering scales from the data.

For the scale selection, we use a selection matrix $\boldsymbol{F} \in \mathbb{R}^{J\times m}$, where each row $\boldsymbol{F}_{(j,\cdot)}, j = 1,\ldots,J$ is dedicated to identifying the diffusion scale of the wavelet $\boldsymbol{P}^{t_j}$ via a one-hot encoding. This is achieved by setting $\boldsymbol{F} := \sigma(\boldsymbol{\Theta}) = [\sigma(\boldsymbol{\theta}_1), \sigma(\boldsymbol{\theta}_2), \ldots, \sigma(\boldsymbol{\theta}_J)]^T$, where $\boldsymbol{\theta}_j \in \mathbb{R}^m$ constitute the rows of the trainable weight matrix $\boldsymbol{\Theta}$, and σ is the softmax function. While this construction may not strictly guarantee an exact one-hot encoding, we assume that the softmax activations yield a sufficient approximation. Further, without loss of generality, we assume that the rows of $\boldsymbol{F}$ are ordered according to the position of the leading "one" activated in every row. In practice, this can be easily enforced by reordering the rows. We now construct the filter bank $\widetilde{\mathcal{W}}_{\boldsymbol{F}} := \{\widetilde{\boldsymbol{\Psi}}_j, \widetilde{\boldsymbol{\Phi}}_J\}_{j=0}^{J-1}$ with the filters

$$\widetilde{\boldsymbol{\Psi}}_0\boldsymbol{x} = \boldsymbol{x} - \sum_{t=1}^{m} \boldsymbol{F}_{(1,t)}\boldsymbol{P}^t\boldsymbol{x}, \quad \widetilde{\boldsymbol{\Phi}}_J\boldsymbol{x} = \sum_{t=1}^{m} \boldsymbol{F}_{(J,t)}\boldsymbol{P}^t\boldsymbol{x}, \quad (5)$$

$$\widetilde{\boldsymbol{\Psi}}_j\boldsymbol{x} = \sum_{t=1}^{m} \left[\boldsymbol{F}_{(j,t)}\boldsymbol{P}^t\boldsymbol{x} - \boldsymbol{F}_{(j+1,t)}\boldsymbol{P}^t\boldsymbol{x}\right], \quad 1 \le j \le J-1,$$

matching and implementing the construction of $\mathcal{W}'_J$ (Eq. 4). While many approaches may be applied to aggregate node-level features into graph-level features such as max, mean, sum pooling, and the more powerful TopK [16] or attention pooling [17], we follow the statistical-moment aggregation explained in Secs. 2-3 motivated by [9, 15] and leave exploration of other pooling methods to future work.

4.1. Incorporating LEGS into a larger neural network

As shown in [9] 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 [18], which we denote LEGS-RBF, to produce the final classification.

The latter configuration more accurately processes scattering features as shown in Table 1. Our RBF network works by first initializing a fixed number of movable anchor points. Then, for every point, new features are calculated based on the radial distances to these anchor points. In previous work on radial basis networks these anchor points were initialized independent of the data. We found that this led to training issues if the range of the data was not similar to the initialization of the centers. Instead, we first use a batch normalization layer to constrain the scale of the features and then pick anchors randomly from the initial features of the first pass through our data. This gives an RBF-kernel network with anchors that are always in the range of the data. Our RBF layer is then $\text{RBF}(\boldsymbol{x}) = \phi(\|\text{BatchNorm}(\boldsymbol{x}) - \boldsymbol{c}\|)$ with $\phi(\boldsymbol{x}) = e^{-\|\boldsymbol{x}\|^2}$.

5. 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 graph diameters for their size. Further, the connectivity patterns of biomolecules are very irregular due to 3D folding and long-range connections, and thus ordinary local node aggregation methods may miss such connectivity differences.

5.1. 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 [9]. We compare against graph convolutional networks (GCN) [2], GraphSAGE [3], graph attention network (GAT) [17], graph isomorphism network (GIN) [4], Snowball network [19], and fixed geometric scattering with a support vector machine classifier (GS-SVM) as in [9], 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 seven biochemical and six social network datasets for graph classification with hundreds to thousands of graphs and tens to hundreds of nodes.

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

Table 1: Mean ± std. over 10 test sets on biochemical (top) and social network (bottom) datasets.

	LEGS-RBF	LEGS-FCN	LEGS-FIXED	GCN	GraphSAGE	GAT	GIN	GS-SVM	Baseline
DD	72.58 ± 3.35	72.07 ± 2.37	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**	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	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	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	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	67.30 ± 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	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	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	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	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	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	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	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

connected than social networks [9]. 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 Tab. 1. On extremely small graphs we find that GS-SVM performs best, which is expected as other methods with more parameters can easily overfit the data. We reason that the performance 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, 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. Additionally, LEGS performs well on social network datasets In Tab. 1, we see that on the social network datasets LEGS performs well. Overlooking the fixed scattering transform GS-SVM, which was tuned in [9] with a focus on these particular social network datasets, a LEGS module architecture is best on three out of the six social datasets and second best on the other three.

Table 2: 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
GIN	221.14 ± 42.56	219.44 ± 34.84
Baseline	393.78 ± 4.02	402.21 ± 21.45

Table 3: Mean ± std. over four runs of mean squared error over 19 targets for the QM9 dataset, lower is better.

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

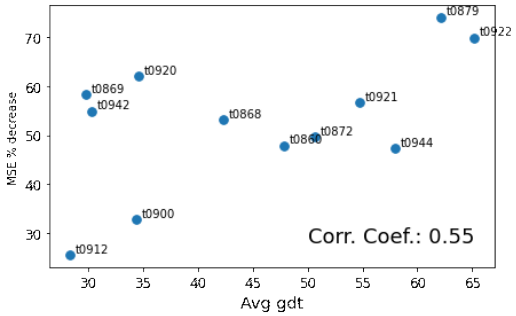

Fig. 2: CASP dataset LEGS-FCN % improvement over GCN in MSE of GDT prediction vs. Average GDT score.

5.2. Graph Regression

We next evaluate learnable scattering on two graph regression tasks, the QM9 [20] graph regression dataset, and a new task from the critical assessment of structure prediction (CASP) challenge [21]. 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 [22]. 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 [21] and preprocess it similarly to [23], 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 nonstructural node features.

LEGS outperforms on all CASP targets. Across all CASP targets we find that LEGS-based architectures significantly outperforms 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. 2 we show the relationship between percent improvement of LEGS over the GCN model and the average GDT score across the target structures. We draw attention to target t0879, where LEGS shows the greatest improvement over other methods. Interestingly this target has particularly long-range dependencies [24]. Since other methods are unable to model these long-range connections, this suggests LEGS is particularly important on these more difficult to model targets.

LEGS outperforms on the QM9 dataset. We evaluate the performance of LEGS-based networks on the quantum chemistry dataset QM9 [20], which consists of 130,000 molecules with $\sim$18 nodes per molecule. We use the node features from [20], 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 and evaluate the mean squared error on the test set with mean and std. over four runs finding that LEGS improves over existing models (Tab. 3).

6. 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.

References

[1] M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Process. Mag.*, 2017.

[2] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *Proc. of ICLR*, 2016.

[3] W. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Adv. in NeurIPS 30*, 2017.

[4] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," *Proc. of ICLR*, 2019.

[5] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. of AAAI*, 2018.

[6] U. Alon and E. Yahav, "On the bottleneck of graph neural networks and its practical implications," in *Proc. of ICLR*, 2021.

[7] P. Barceló, E. Kostylev, M. Monet, J. Pérez, J. Reutter, and J. Silva, "The logical expressiveness of graph neural networks," in *Proc. of ICLR*, 2020.

[8] S. Mallat, "Group invariant scattering," *Commun. Pure Appl. Math.*, 2012.

[9] F. Gao, G. Wolf, and M. Hirn, "Geometric scattering for graph data analysis," in *Proc. of ICML*, 2019.

[10] F. Gama, A. Ribeiro, and J. Bruna, "Diffusion scattering transforms on graphs," in *Proc. of ICLR*, 2019.

[11] D. Zou and G. Lerman, "Graph convolutional neural networks via scattering," *Appl. Comp. Harm. Anal.*, 2019.

[12] M. Perlmutter, G. Wolf, and M. Hirn, "Geometric scattering on manifolds," in *NeurIPS 2018 Workshop on Integration of Deep Learning Theories*, 2018.

[13] R. Coifman and M. Maggioni, "Diffusion wavelets," *Appl. Comp. Harm. Anal.*, vol. 21(1), pp. 53–94, 2006.

[14] F. Gama, J. Bruna, and A. Ribeiro, "Stability of graph scattering transforms," *Adv. in NeurIPS 32*, 2019.

[15] M. Perlmutter, F. Gao, G. Wolf, and M. Hirn, "Understanding graph neural networks with asymmetric geometric scattering transforms," arXiv:1911.06253, 2019.

[16] H. Gao and S. Ji, "Graph U-Nets," in *Proc. of ICML*, 2019, vol. 97 of *PMLR*, pp. 2083–2092.

[17] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *Proc. of ICLR*, 2018.

[18] D. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, vol. 2, pp. 321–355, 1988.

[19] S. Luan, M. Zhao, X. Chang, and D. Precup, "Break the ceiling: Stronger multi-scale deep graph convolutional networks," in *Adv. in NeurIPS 32*, 2019.

[20] J. Gilmer, S. Schoenholz, P. Riley, O. Vinyals, and G. Dahl, "Neural message passing for quantum chemistry," in *Proc. of ICML*, 2017, vol. 70 of *PMLR*, pp. 1263–1272.

[21] J. Moult, K. Fidelis, A. Kryshtafovych, T. Schwede, and A. Tramontano, "Critical assessment of methods of protein structure prediction (CASP)—Round XII," *Proteins Struct. Funct. Bioinforma.*, 2018.

[22] V. Modi, Q. Xu, S. Adhikari, and R. Dunbrack, "Assessment of Template-Based Modeling of Protein Structure in CASP11," *Proteins*, 2016.

[23] J. Ingraham, V. Garg, R. Barzilay, and T. Jaakkola, "Generative models for Graph-Based Protein Design," in *Adv. in NeurIPS 32*, 2019.

[24] S. Ovchinnikov, H. Park, D. Kim, F. DiMaio, and D. Baker, "Protein structure prediction using Rosetta in CASP12," *Proteins Struct. Funct. Bioinforma.*, 2018.