Multiresolution Graph Transformers and Wavelet Positional Encoding for Learning Long-Range and Hierarchical Structures

Nhat Khang Ngo,[1,2] Truong Son Hy,[1,3] and Risi Kondor[4]
[1)] *Equal contribution*
[2)] *FPT Software AI Center, Hanoi 10000, Vietnam*
[3)] *Halıcıoğlu Data Science Institute, University of California San Diego, La Jolla, CA 92093, USA*
[4)] *Department of Computer Science, University of Chicago, Chicago, IL 60637, USA*

(*Electronic mail: khangnn4@fpt.com, tshy@ucsd.edu, risi@uchicago.edu)

(Dated: 26 June 2023)

Contemporary graph learning algorithms are not well-suited for large molecules since they do not consider the hierarchical interactions among the atoms, which are essential to determine the molecular properties of macromolecules. In this work, we propose Multiresolution Graph Transformers (MGT), the first graph transformer architecture that can learn to represent large molecules at multiple scales. MGT can learn to produce representations for the atoms and group them into meaningful functional groups or repeating units. We also introduce Wavelet Positional Encoding (WavePE), a new positional encoding method that can guarantee localization in both spectral and spatial domains. Our proposed model achieves competitive results on three macromolecule datasets consisting of polymers, peptides, and protein-ligand complexes, along with one drug-like molecule dataset. Significantly, our model outperforms other state-of-the-art methods and achieves chemical accuracy in estimating molecular properties (e.g., GAP, HOMO and LUMO) calculated by Density Functional Theory (DFT) in the polymers dataset. Furthermore, the visualizations, including clustering results on macromolecules and low-dimensional spaces of their representations, demonstrate the capability of our methodology in learning to represent long-range and hierarchical structures. Our PyTorch implementation is publicly available at `https://github.com/HySonLab/Multires-Graph-Transformer`.

I. INTRODUCTION

Macromolecules are long-range and hierarchical structures as they consist of many substructures. While small molecules in existing datasets[1–3] comprise less than 50 atoms connected by simple rings and bonds, this number in a macromolecule can be dozens or even hundreds. Substructures such as repeating units and functional groups are intrinsic parts of macromolecules; they present unique chemical reactions regardless of other compositions in the same molecules[4]. Therefore, studying the multiscale, i.e. multiresolution, characteristic of large molecules is imperative to gain comprehensive knowledge about real-life materials like polymers or proteins[5]. In recent years, several works[6–8] have been proposed to apply machine learning algorithms to learn macromolecules at multiple scales. These approaches, however, rely on thorough feature selection and extraction, which are not efficient when learning from large databases of materials[6].

Message passing is a prevailing paradigm for designing neural networks that operate on graph-structured data. Previous studies[9–13] have proposed different strategies to perform message passing on graphs and achieved remarkable results across various domains. However, message-passing-dominated graph neural networks (GNNs) have some limitations, such as limited expressiveness capability[13,14], over-smoothing[15–17], over-squashing[18] issues. Over-smoothing exists in graph neural networks that consist of a sufficiently large number of layers, and node representations are likely to converge to a constant after going through these deep networks. Over-squashing problems occur when messages are ineffectively propagated and aggregated through bottlenecks on long-range graph structures. These two shortcomings hinder GNNs from making good predictions on long-range and hierarchically structured data. Furthermore, the molecular properties of large molecules are formed not only by interactions among atoms within neighborhoods but also by distant atoms. Therefore, local information is not sufficient to model macromolecules.

Transformers are classes of deep learning models that leverage self-attention mechanisms to handle long-range dependencies in various data domains, such as natural language processing[19,20] or computer vision[21,22]. In graph domains, Transformer-like architectures[23–26] have proved their effectiveness in learning node representations as they can overcome the over-smoothing and over-squashing issues by directly measuring the pairwise relationships between the nodes. Contrary to GNNs, graph transformers do not use the graph structure as hard-coded information. They, instead, encode positional and structural information on graphs as soft inductive bias, making them flexible learners in graph learning problems[23,27]. Node positional representations can be derived based on spectral[28,29] or spatial[30,31] domains. Most existing spectral-based methods decompose the graph Laplacian into sets of eigenvectors and eigenvalues. However, these eigenvectors have sign ambiguity and are unstable due to eigenvalue multiplicities[32]. On the other hand, spatial-based approaches compute the shortest distances among the nodes; however, these encoding methods do not consider the structural similarity between nodes and their neighborhoods[33].

We propose Multiresolution Graph Transformer (MGT) and Wavelet Positional Encoding (WavePE), using multiresolution analysis on both spectral and spatial domains for learning to represent hierarchical structures. Our contributions are four-fold:

- We design Multiresolution Graph Transformer (MGT), a Transformer-like architecture that can operate on macromolecules at multiple scales. Our proposed model can learn the atomic representations and group them into meaningful clusters via a data-driven algorithm. Finally, the substructures, i.e. clusters, are fed to a Transformer encoder to calculate the representations of several substructures in macromolecules.
- We introduce Wavelet Positional Encoding (WavePE), a new positional encoding scheme for graph-structured data. Since wavelet analysis can provide localization in both spatial and spectral domains, we construct a set of wavelets to capture the structural information on graphs at different scales. Then, we apply equivariant encoding methods to project the wavelet tensors into positional representations for the atoms.
- We show the effectiveness of our methodology by reporting its superior performance on three molecular property prediction benchmarks. These datasets contain macromolecules, i.e. peptides and polymers, that are highly hierarchical and consist of up to hundreds of atoms. Our model achieves the important chemical accuracy in DFT approximation for the polymers dataset.
- Our visualization demonstrates the comprehensiveness of our proposed methods in learning to represent large molecules. In general, we show the representations of molecules produced by MGT and how MGT determines and groups the atoms in long-chain molecules.

II. RELATED WORK

a. Hierachical Learning on Molecules Functional groups or repeating units are essential phenomena in chemistry. While functional groups constitute large molecules, repeating units are the primary parts that produce complete polymer chains. We regard them as substructures. In particular, similar substructures undergo similar chemical reactions regardless of the remaining compositions existing in the molecules[4]. Previous work has leveraged the hierarchical property of molecules to improve the performance in molecular representation learning and generation. Fang *et al.* [34], Rong *et al.* [35], and Chen, Park, and Park [36] use functional groups as prior knowledge to guide the models to predict accurate molecular properties. For the molecular generation task, Maziarz *et al.* [37] and Jin, Barzilay, and Jaakkola [38] follow several chemical rules to extract substructures and construct a vocabulary of structural motifs to generate large molecules.

b. Graph Transformers Earlier research efforts have adopted Transformer-like architectures to graph-structured data. Dwivedi and Bresson [28] proposed an early approach to generalize Transformers to graphs using Laplacian positional encoding and performing self-attention on one-hop neighbors surrounding center nodes. On the other hand, Kreuzer *et al.* [23] compute attention scores on the entire graph with differentiation between positive and negative edges, while also using Laplacian positional encoding. Rong *et al.* [35] introduce GTransformer that utilizes vectorized outputs from local GNNs as inputs for a Transformer encoder, making up an effective combination between node local and global information. Rampášek *et al.* [25] propose a general framework that integrates essential components of Graph Transformers, including positional or structural encoding, graph feature extraction, local message passing, and self-attention. Also, Chen, O'Bray, and Borgwardt [33] extract multiple k-hop subgraphs and feed them to local GNNs to compute their embeddings, which are then moved to a Transformer encoder. Graphormer proposed in[39] use attention mechanisms to estimate several types of encoding, such as centrality, spatial, and edge encodings. In addition, Kim *et al.* [27] treat all nodes and edges as independent tokens augmented with orthonormal node identifiers and trainable type identifiers and fed them to a standard Transformer encoder. Moreover, Yun *et al.* [40] generate multiple meta-paths, i.e. views, of a graph and computed their pairwise attention scores, before aggregating them into a final representation for the entire graph. Cai *et al.* [26] has analyzed the theoretical relationship between graph transformers and the conventional message passing neural networks.

c. Graph Positional Encoding Several approaches have been proposed to encode the positional or structural representations into node features to improve the expressiveness of GNNs and Graph Transformers. Node positions can be determined via spectral or spatial domains. Spectral-based methods include Laplacian positional encoding[23,28] and random walk positional encoding (RWPE)[29]. For spatial-based methods, You, Ying, and Leskovec [30] compute distances of sets of nodes to anchor nodes, whereas Li *et al.* [31] calculate the shortest distances between pairs of nodes.

d. Multiresolution Analysis and Wavelet Theory Multiresolution Analysis (MRA) has been proposed by[41,42] as a method to approximate signals at multiple scales in which the signals are decomposed over elementary waveforms chosen from a family called wavelets (i.e. mother wavelets and father wavelets), including Haar[43], Daubechies[44], etc., to produce the sparse representations. In graph and discrete domains, Hammond, Vandergheynst, and Gribonval [45] introduced spectral graph wavelets that are determined by applying the wavelet operator on the graph Laplacian at multi-levels. Coifman and Maggioni [46] propose diffusion wavelet that is a fast multiresolution framework for analyzing functions on discretized structures such as graphs and manifolds. In the deep learning era, Rustamov and Guibas [47] and Xu *et al.* [48] leverage the power of neural networks for graph wavelet construction and computation. Hy and Kondor [49] proposed a learning algorithm to construct graph wavelets via multiresolution matrix factorization[50].

III. BACKGROUND

A. Notation

A molecule can be represented as an undirected graph in which nodes are the atoms and edges are the valency bonds between them. In particular, we refer to a molecular graph

as $G = (\mathrm{V}, \mathrm{E}, A, X_v, X_e, P, \mathrm{V}_s)$, where G is an undirected graph having $\mathrm{V} = \{v_1, v_2, \cdots, v_n\}$ (let $n = |\mathrm{V}|$ denote the number of nodes) and $E = \{(u,v)|u,v \in \mathrm{V}\}$ (let $n_e = |\mathrm{E}|$ denote the number of edges) as sets of nodes and edges respectively; also, $A \in \mathbb{R}^{n\times n}$ is the graph's adjacency matrix. When a graph is attributed, we augment G with a matrix arrangement of node feature vectors $X_v = (x_1, ..., x_n)^{\mathrm{T}}, x_i \in \mathbb{R}^{d_v}$ and node positional vectors $P = (p_1, ..., p_n)^{\mathrm{T}}, p_i \in \mathbb{R}^{p}$. Furthermore, the covalent bonds between atoms are regarded as edge features and stored in $X_e = (e_1, ..., e_{n_e})^{\mathrm{T}}, e_i \in \mathbb{R}^{d_e}$. In addition to the atom-level representation of G, $\mathrm{V}_s = \{v_{s_1}, ..., v_{s_k}\}$ denotes the substructure set in which $v_{s_i} \subset \mathrm{V}$, i.e. v_{s_i} is a subset of atoms of the molecule. In Section III C, we define methods to produce features of substructure-level nodes by leveraging the hierarchical properties of macromolecules.

B. Message-passing Neural Networks

We model the local interactions of atoms using a well-known class of graph neural networks (GNNs) known as message-passing neural networks (MPNNs)[51]. In the context of modeling molecules, atom nodes communicate with each other in a local neighborhood by exchanging vectorized messages. These messages are updated using neural networks. In particular, the model considers the interactions between neighboring atoms by modeling the propagation of atomic features along the covalence bonds in the molecular graph. With hidden embeddings h_u representing each node $u \in V$, we define GNN message-passing mechanism based on a general framework introduced in [51] as

$$m_u^{(l)} = \sum_{v\in\mathscr{N}(u)} \mathbf{M}_l(h_u^{(l-1)}, h_v^{(l-1)}, e_{vu}^{(l-1)}), \tag{1}$$

$$h_u^{(l)} = \mathbf{U}_l(h_u^{(l-1)}, m_u^{(l)}), \tag{2}$$

$$e_{vu}^{(l)} = \mathbf{U}_l^e(e_{vu}^{(l-1)}, m_u^{(l)}). \tag{3}$$

This framework propagates messages on graph G through L layers. At the l^{th} layer, vectorized messages $m_u^{(l)}$ are computed based on message functions $\mathbf{M}_l$ whose inputs involve three arguments, including $h_u^{(l-1)}$ and $h_v^{(l-1)}$ are node embeddings of u and its neighbors $v \in \mathscr{N}(u)$, and $e_{vu}^{(l-1)}$ indicates (vectorized) edge features between them at the $(l-1)^{th}$ layer. Then, the messages are passed to node update functions $\mathbf{U}_l$ that calculate new node embeddings $h_u^{(l)}$ for node u using its previous embedding $h_u^{(l-1)}$ and the messages $m_u^{(l)}$. In addition, h_u^0 equals $x_u \in X_v$, the initial atom features. Besides node embeddings, some MPNNs[11,52] use an additional update function $\mathbf{U}_l^e$ to compute new edge embeddings $e_{vu}^{(l)} = \mathbf{U}_l^e(e_{vu}^{(l-1)}, m_u^{(l)})$. In general, we write a layer of MPNN that exchange the messages and updates both node and edge embeddings as

$$H^{(l)}, E^{(l)} = \mathrm{MPNN}_l(H^{(l-1)}, E^{(l-1)}, A), \tag{4}$$

where $H^{(l)}$ and $E^{(l)}$ denote node and edge embeddings at the l^{th} layer.

For example, **graph convolution network (GCN)**[10] can be regarded as an MPNN with the message at node u is $m_u^{(l)} = \sum_{v\in N(u)\cup\{u\}} \frac{e_{vu}}{\sqrt{|\mathscr{N}(v)||\mathscr{N}(u)|}} h_v^{(l-1)}$, which aggregates the normalized information of neighborhood embeddings multiplied with the edge weights. A new embedding of node u is computed using the update function $h_u^l = \sigma(W_l m_u^{(l)})$, where W_l is a trainable parameter matrix of layer l, and σ denotes an element-wise nonlinearity (e.g., ReLU, tanh, sigmoid, etc.). To account for the vectorization of node embeddings, we can rewrite a layer of GCN as: $H^{(l)} = \sigma(\widetilde{A}H^{(l-1)}W_l)$, where $H^{(0)} = X_u$, and $\widetilde{A} = \widetilde{D}^{-1/2}(A+I)\widetilde{D}^{-1/2}$, where $\widetilde{D}_{ii} = \sum_j A_{ij}$ denotes the diagonal degree matrix.

Node and edge embeddings at the L^{th} layer can be used for predicting node-level and edge-level properties. For graph-level predictions, some readout function $\mathbf{R}$ is used to aggregate all node embeddings to produce an embedding vector z for the entire graph G as

$$z = \mathbf{R}(\{h_u^{(\mathrm{L})} | u \in \mathrm{V}\}), \tag{5}$$

where $h_u^{(\mathrm{L})}$ denotes the embedding vector of node u at the final L^{th} layer. Also, $\mathbf{R}$ must be invariant with respect to permutations and differentiable.

C. Hierachical Learning on Molecules

Molecular property prediction is regarded as a graph-level learning task. We need to aggregate node embeddings into graph-level vectors which are then fed to a classifier to make predictions on graphs. Specifically, a function $f : \mathrm{V} \to \mathbb{R}^{d_o}$ that maps the atom $u \in \mathrm{V}$ to a d_o-dimensional vector $z_u \in \mathbb{R}^{d_o}$ should learn to produce atom-level embeddings. Most existing graph neural networks compute the vector $z = \mathbf{R}(\{f(u)|u \in \mathrm{V}\})$ that indicates an embedding for the entire molecular graph, where $\mathbf{R}$ can be sum, mean, max, or more sophisticated operators. For hierarchical learning, substructure-level representations can be derived in addition to atom-level representations by aggregating node representations in the same substructures as $z_s = \mathbf{R}(\{f(u)|u \in v_s \wedge v_s \in \mathrm{V}_s\})$. Instead of atom vectors, we aggregate the substructure vectors to represent the entire graph, i.e. $z = \mathbf{R}(\{z_s|z_s \in \mathrm{V}_s\})$. Finally, a classifier g given z as inputs is trained to predict the molecular properties.

D. Transformers on Graphs

While GNNs learn node embeddings by leveraging the graph structure via local message-passing mechanisms, Transformers disregard localities and directly infer the relations between pairs of nodes using only node attributes. In other words, the node connectivity is not utilized in pure transformer-like architectures[19]; as a result, the problem is simplified to learning on sets. Given a matrix of node features $X \in \mathbb{R}^{n\times d}$, Transformers compute three matrices including query (Q), key

(K), and value (V) via three linear transformations $Q = XW_q^T$, $K = XW_k^T$, and $V = XW_v^T$. A self-attention matrix (H) can be computed as follows:

$$H = \text{softmax}\left(\frac{QK^T}{\sqrt{d_o}}\right)V, \tag{6}$$

where W_q, W_k, and W_v are learnable parameters in $\mathbb{R}^{d_o \times d}$, resulting in $H \in \mathbb{R}^{n \times d_o}$. Furthermore, H in Eq. 6 denotes an attention head. To improve effectiveness, multiple $\{H\}_{i=1}^h$ are computed, which is known as multi-head attention. All of the attention heads are concatenated to form a final tensor: $H_o = \text{concat}(H_1, ..., H_h)$, where h is the number of attention heads. Finally, the output X', i.e. new node embeddings, can be computed by feeding H_o into a feed-forward neural network (FFN), i.e. $X' = \text{FFN}(H_o)$. It is easy to see that Transformers operating on inputs without positional encoding are permutation invariant.

E. Spectral Positional Encoding

As pure Transformer encoders only model the global interactions between nodes without being cognizant of the graph structures, node features should be augmented by positional features to preserve the structural information, which is critical in graph learning. For spectral positional encodings, each node u of a weighted graph G having n nodes (see Section III A) is assigned to a point $p_u \in \mathbb{R}^m$ ($m << n$). Here, given the adjacency matrix A, the mapping from u to p_u should minimize the "energy" $\mathscr{E}$:

$$\mathscr{E} = \sum_{(u,v)\in E} \|p_u - p_v\|^2 A_{uv}. \tag{7}$$

The intuition behind this is that if two neighboring nodes u and v are "close", then their positional vectors in the m-dimensional space are close as well. Belkin and Niyogi[53] shows that the solution is provided by the matrix of eigenvectors that correspond to m smallest non-trivial eigenvalues of $L = D - A$, the Laplacian matrix of graph G, and $D_{ii} = \sum_j A_{ij}$.

In particular, to compute node positional features, L are eigendecomposed first as:

$$L = U\Sigma U^T \tag{8}$$

here, $\Sigma = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_n)$ is the diagonal matrix of n eigenvalues that are $0 = \lambda_1 < \lambda_2 \leq \lambda_3 \leq ... \leq \lambda_n$, and $U = (u_1, u_2, ..., u_n)^T$, $u_i \in \mathbb{R}^n$, denotes the matrix of their corresponding orthogonal eigenvectors, which form a graph Fourier basis. Then, except for λ_1, m smallest eigenvalues (i.e. $\lambda_2 \leq \lambda_3 \leq ... \leq \lambda_m$) are chosen with their associated eigenvectors to construct a sub-matrix $P \in \mathbb{R}^{n \times m}$ of U. In particular, every row in matrix P corresponds to the first m values of the respective row in matrix U. The matrix P serves as a node positional feature, which can be combined or concatenated with the node atomic features X_v, producing comprehensive inputs for subsequent Transformers. Figure 1 illustrates this spectral encoding scheme on the Aspirin $C_9H_8O_4$ molecular graph.

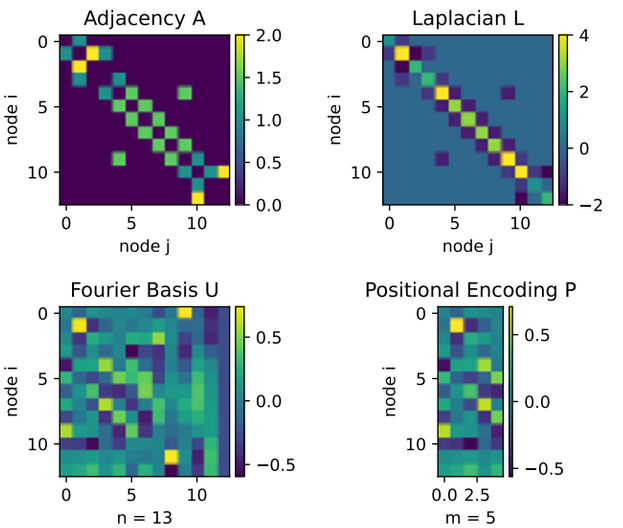


FIG. 1: Visualizations of graph adjacency A, graph Laplacian L, Fourier basis U, and positional encoding feature P on the molecular graph of Aspirin $C_9H_8O_4$ with 13 nodes (i.e. heavy atoms).

IV. WAVELET POSITIONAL ENCODING

A. Spectral Graph Wavelets

Let $A \in \mathbb{R}^{n \times n}$ be the adjacency matrix of an undirected graph $G = (V, E)$. The normalized graph Laplacian is defined as $L_{\text{norm}} = I_n - \widetilde{D}^{-1/2} A \widetilde{D}^{-1/2}$, where I_n is the identity matrix and $\widetilde{D}$ is the diagonal matrix of node degrees as mentioned in Section III B. L_{norm} can be decomposed into a complete set of orthonormal eigenvectors $U = (u_1, u_2, ..., u_n)$ associated with real and non-negative eigenvalues $\{\lambda\}_1^n$. While graph Fourier transform uses U as a set of bases to project the graph signal from the vertex domain to the spectral domain, graph wavelet transform constructs a set of spectral graph wavelets as bases for this projection via:

$$\psi_s = U\Sigma_s U^T$$

where $\Sigma_s = \text{diag}(g(s\lambda_1), g(s\lambda_2), ..., g(s\lambda_n))$ is a scaling matrix of eigenvalues, $\psi_s = (\psi_{s1}, \psi_{s2}, ..., \psi_{sn})$ and each wavelet ψ_{si} indicates how a signal diffuses away from node i at scale s; we choose $g(s\lambda) = e^{-s\lambda}$ as a heat kernel[54]. Since a node's neighborhoods can be adjusted by varying the scaling parameter s[55], using multiple sets of wavelets at different scales can provide comprehensive information on the graph's structure. It means that larger values of s_i correspond to larger neighborhoods surrounding a center node. Figure 2 illustrates how wavelets can be used to determine neighborhoods at different scales on a molecular graph. In this work, we leverage this property of graph wavelets to generate node positional representations that can capture the structural information of a center node on the graph at different resolutions. We employ k diffusion matrices $\{\psi_{s_i}\}_{i=1}^k$ in which each ψ_{s_i} has a size

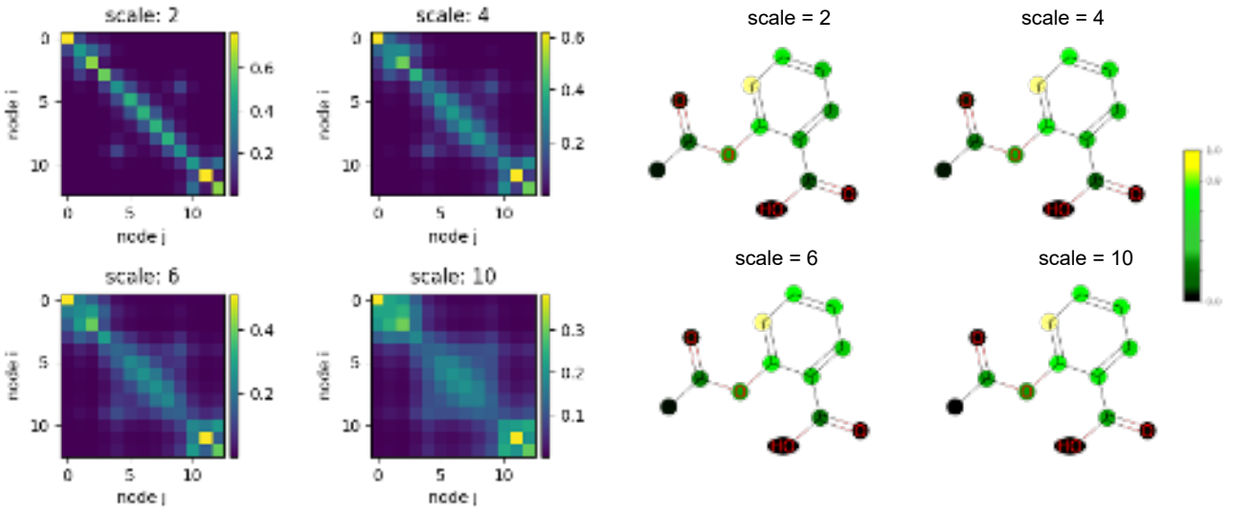


FIG. 2: Visualization of some of the wavelets with scaling parameters on the Aspirin $C_9H_8O_4$ molecular graph with 13 nodes (i.e. heavy atoms). The center node is colored yellow. The colors varying from bright to dark illustrate the diffusion rate from the center node to the others, i.e. nodes that are closer to the center node have brighter colors. Low-scale wavelets are highly localized, whereas the high-scale wavelets can spread out more nodes on the molecular graphs

of $n \times n$, resulting in a tensor of graph wavelets $\mathbf{P} \in \mathbb{R}^{n\times n\times k}$. Additionally, WavePE is a generalized version of RWPE[29] as the random walk process can be regarded as a type of discrete diffusion. In the following section, we demonstrate the use of tensor contractions to generate a matrix of node positional representations $P \in \mathbb{R}^{n\times k}$ from $\mathbf{P}$. In general, Figures 3 and 4a demonstrate our wavelet positional encoding method.

B. Equivariant Encoding

It is important to note that our spectral graph wavelets computed from the previous section must be further encoded in a permutation-equivariant manner. For simplicity, that means if we permute (i.e. change the order) the set of nodes, their position encodings must be transformed accordingly. In this section, we formally define permutation symmetry, i.e. symmetry to the action of the symmetric group, $\mathbb{S}_n$, and construct permutation-equivariant neural networks to encode graph wavelets. An element $\sigma \in \mathbb{S}_n$ is a permutation of order n, or a bijective map from $\{1,\dots,n\}$ to $\{1,\dots,n\}$. For example, the action of $\mathbb{S}_n$ on an adjacency matrix $A \in \mathbb{R}^{n\times n}$ and on a latent matrix (i.e. node embedding matrix) $Z \in \mathbb{R}^{n\times d_z}$ are:

$$[\sigma\cdot A]_{i_1,i_2} = A_{\sigma^{-1}(i_1),\sigma^{-1}(i_2)}, \quad [\sigma\cdot Z]_{i,j} = Z_{\sigma^{-1}(i),j},$$

for $\sigma \in \mathbb{S}_n$. Here, the adjacency matrix A is a second-order tensor with a single feature channel, while the latent matrix Z is a first-order tensor with d_z feature channels. In general, the action of $\mathbb{S}_n$ on a k-th order tensor $X \in \mathbb{R}^{n^k\times d}$ (i.e. the last index denotes the feature channels) is defined similarly as:

$$[\sigma\cdot X]_{i_1,..,i_k,j} = X_{\sigma^{-1}(i_1),..,\sigma^{-1}(i_k),j}, \qquad \sigma \in \mathbb{S}_n.$$

Formally, we define these equivariant and invariant properties in Def. IV.1 and equivariant neural networks in Def. IV.2.

Definition IV.1. An $\mathbb{S}_n$-equivariant (or permutation equivariant) function is a function $f\colon \mathbb{R}^{n^k\times d} \to \mathbb{R}^{n^{k'}\times d'}$ that satisfies $f(\sigma\cdot X) = \sigma\cdot f(X)$ for all $\sigma \in \mathbb{S}_n$ and $X \in \mathbb{R}^{n^k\times d}$. Similarly, we say that f is $\mathbb{S}_n$-invariant (or permutation invariant) if and only if $f(\sigma\cdot X) = f(X)$.

Definition IV.2. An $\mathbb{S}_n$-equivariant network is a function $f : \mathbb{R}^{n^k\times d} \to \mathbb{R}^{n^{k'}\times d'}$ defined as a composition of $\mathbb{S}_n$-equivariant linear functions $f_1,..,f_T$ and $\mathbb{S}_n$-equivariant nonlinearities $\gamma_1,..,\gamma_T$:

$$f = \gamma_T \circ f_T \circ .. \circ \gamma_1 \circ f_1.$$

On the another hand, an $\mathbb{S}_n$-invariant network is a function $f : \mathbb{R}^{n^k\times d} \to \mathbb{R}$ defined as a composition of an $\mathbb{S}_n$-equivariant network f' and an $\mathbb{S}_n$-invariant function on top of it, e.g., $f = f''\circ f'$.

In order to build permutation-equivariant neural networks, we revisit some basic tensor operations: the tensor product $A\otimes B$ (see Def. IV.3) and tensor contraction $A_{\downarrow_{x_1,..,x_p}}$ (see Def. IV.4). It can be shown that these tensor operations respect permutation equivariance[56,57].

Definition IV.3. The **tensor product** of $A \in \mathbb{R}^{n^a}$ with $B \in \mathbb{R}^{n^b}$ yields a tensor $C = A\otimes B \in \mathbb{R}^{n^{a+b}}$ where

$$C_{i_1,i_2,..,i_{a+b}} = A_{i_1,i_2,..,i_a} B_{i_{a+1},i_{a+2},..,i_{a+b}}.$$

Definition IV.4. The **contraction** of $A \in \mathbb{R}^{n^a}$ along the pair of dimensions $\{x,y\}$ (assuming $x < y$) yields a $(a-2)$-th order

tensor

$$C_{i_1,..,i_{x-1},j,i_{x+1},..,i_{y-1},j,i_{y+1},..,i_a} = \sum_{i_x=i_y} A_{i_1,..,i_a},$$

where we assume that i_x and i_y have been removed from amongst the indices of C. Using Einstein notation, this can be written more compactly as

$$C_{\{i_1,i_2,..,i_a\}\setminus\{i_x,i_y\}} = A_{i_1,i_2,..,i_a}\delta^{i_x,i_y},$$

where δ is the Kronecker delta. In general, the contraction of A along dimensions $\{x_1,..,x_p\}$ yields a tensor $C = A_{\downarrow x_1,..,x_p} \in \mathbb{R}^{n^{a-p}}$ where

$$A_{\downarrow x_1,..,x_p} = \sum_{i_{x_1}}\sum_{i_{x_2}}...\sum_{i_{x_p}} A_{i_1,i_2,..,i_a},$$

or compactly as

$$A_{\downarrow x_1,..,x_p} = A_{i_1,i_2,..,i_a}\delta^{i_{x_1},i_{x_2},..,i_{x_p}}.$$

Based on these tensor contractions and Def. IV.1, we can construct the second-order $\mathbb{S}_n$-equivariant networks encoding a graph with adjacency matrix $A \in \mathbb{R}^{n\times n}$, node features $X_v \in \mathbb{R}^{n\times d_v}$ and edge features $X_e \in \mathbb{R}^{n\times n\times d_e}$ as in Section IV B 1:

$$f = \gamma\circ M_T \circ .. \circ \gamma \circ M_1.$$

The "raw" graph wavelets can be treated as a second-order tensor of size $n\times n\times k$ where k is the number of scales, similarly as the edge features. We employ the higher-order permutation-equivariant message passing proposed by Maron *et al.* [58], Hy *et al.* [56] and Kondor *et al.* [57] to encode the "raw" graph wavelets from size $n\times n\times k$ into $n\times k$ that will be further used as nodes/tokens' embeddings of our Transformer architecture (see Figures 3 and 4).

1. Second-order message passing

The second order message passing has the embedding matrix $H_0 \in \mathbb{R}^{|\mathrm{V}|\times|\mathrm{V}|\times(d_v+d_e)}$ initialized by promoting the node features X_v to a second order tensor (e.g., we treat node features as self-loop edge features), and concatenating with the edge features X_e. Iteratively,

$$H_t = \gamma(M_t), \quad M_t = W_t\left[\bigoplus_{i,j}\downarrow(A\otimes H_{t-1})_{i,j}\right], \qquad (9)$$

where $A\otimes H_{t-1}$ results in a fourth order tensor while $\downarrow_{i,j}$ contracts it down to a second order tensor along the i-th and j-th dimensions, $\oplus$ denotes concatenation along the feature channels, and W_t denotes a multilayer perceptron on the feature channels. We remark that the popular MPNNs[51] is a lower-order one and a special case in which $M_t = \widetilde{D}^{-1}AH_{t-1}W_{t-1}$ where $\widetilde{D}_{ii} = \sum_j A_{ij}$ is the diagonal matrix of node degrees (see Section III B). The embedding matrix H_T of the last iteration is still second order, so we contract it down to the first order latent $Z = \bigoplus_i \downarrow H_{Ti}$.

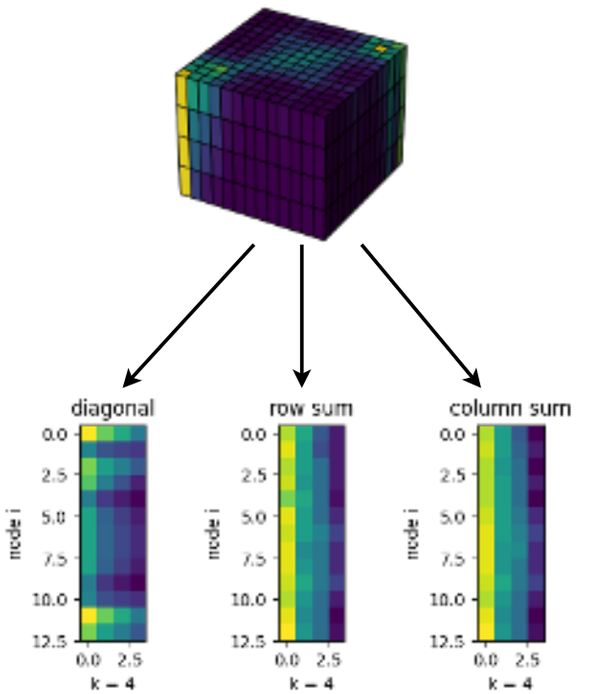


FIG. 3: Demonstration of three basic tensor contractions (i.e. diagonal, row sum, and column sum) on a second-order tensor of size $n\times n\times k$ that is resulted from concatenating the "raw" graph wavelets at k different scales (see Fig. 2). In this case of Aspirin molecule, $n = 13$ and $k = 4$.

V. MULTIRESOLUTION GRAPH TRANSFORMERS

In this section, we present Multiresolution Graph Transformers (MGT), a neural network architecture for learning hierarchical structures. MGT uses Transformers to yield the representations of macromolecules at different resolutions. While previous work either neglects the hierarchical characteristics of large molecules or fails to model global interactions between distant atoms, our proposed approach can satisfy these two properties via multiresolution analysis. Figs. 4b and 4c show an overview of our framework. MGT consists of three main components: an atom-level encoder, a module to extract substructures, and a substructure-level encoder. We use a graph transformer to generate the atomic embeddings. Then, substructures present in molecules are extracted by a learning-to-cluster algorithm. The molecular graph is coarsened into a set of substructures, and we use a pure Transformer encoder to learn their relations.

A. Atom-Level Encoder

To utilize the proposed wavelet positional encoding demonstrated in Section IV, we leverage the design of the graph transformer proposed by Rampášek *et al.* [25], which is a

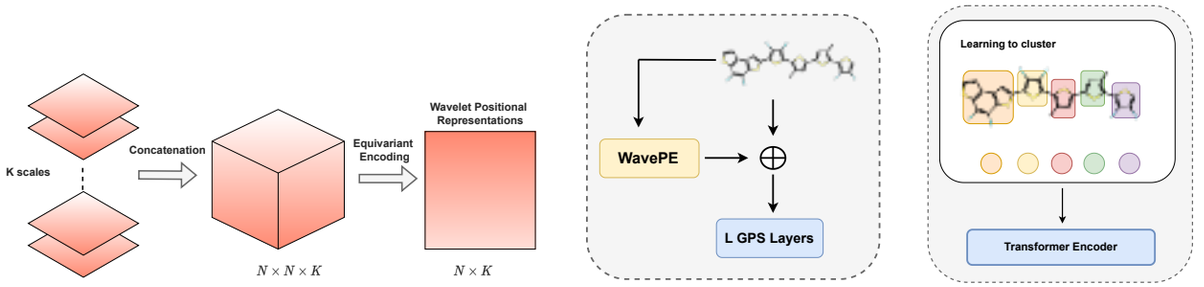


FIG. 4: Overview of Wavelet Positional Encoding (WavePE) and Multiresolution Graph Transformer (MGT). a) k diffusion matrices of size $N \times N$ are stacked together to produce a wavelets tensor with size $N \times N \times K$ which are contracted by equivariant encoding methods to yield a tensor of positional representation $N \times k$. b) Atomic representations are derived by passing the molecular graph augmented with positional features through L layers of GPS (i.e. scalable Graph Transformer proposed in [25]). c) A macromolecule is decomposed into several substructures in which the features are aggregated from the atom-level outputs, resulting in a set of substructures that are moved to a Transformer encoder.

general, powerful, and scalable Graph Transformer (GPS) for graph representation learning. Let $A \in \mathbb{R}^{n\times n}$ be the adjacency matrix of a graph with n nodes and n_e edges; $X^{(l)}$ and $E^{(l)}$ are node and edge features at layer l^{th}, respectively. In addition, $X^{(0)} \in \mathbb{R}^{n\times d}$ and $E^{(0)} \in \mathbb{R}^{n_e\times d}$ are initial atom and bond features embedded in d-dimensional spaces created by two embedding layers. The wavelet positional vectors $p \in \mathbb{R}^{n\times k}$ are fed to an encoder (e.g., a feed-forward neural network or a linear transformation), yielding a matrix of positional features $P \in \mathbb{R}^{n\times d_p}$. We let $X^{(0)} = \text{concat}(X^{(0)}, P)$ to produce new node features $X^{(0)} \in \mathbb{R}^{n\times(d+d_p)}$. From here, we define $d = d + d_p$, and for convenience, the output dimensions of all layers are equal to d.

Each layer of GPS uses a message-passing neural network (MPNN_l) to exchange messages within the neighborhood and a self-attention layer (SA_l) described in Equation 6 to compute global interactions among distant nodes:

$$X^{(l)}_{\text{loc}}, E^{(l)} = \text{MPNN}_l(X^{(l-1)}, E^{(l-1)}, A), \tag{10}$$

$$X^{(l)}_{\text{glob}} = \text{SA}_l(X^{(l-1)}), \tag{11}$$

$$X^{(l)} = \text{FFN}_l(X^{(l)}_{\text{loc}} + X^{(l)}_{\text{glob}}), \tag{12}$$

where $X^{(l)}_{\text{loc}}$ and $X^{(l)}_{\text{glob}}$ are node local and global embeddings; they are unified into $X^{(l)}$ via Eq. 12. Popular techniques such as Dropout[59] and normalization[60,61] are omitted for the sake of clarity. By feeding the molecular graph through L layers, we obtain two matrices $X_a = X^{(\text{L})}$ and $E_a = E^{(\text{L})}$ indicating the atom-level node and edge embeddings, respectively.

B. Learning to Cluster

In this work, we use a message-passing neural network augmented with differentiable pooling layers[62,63] to cluster the atoms into substructures automatically:

$$Z = \text{MPNN}_{\text{e}}(X_a, E_a, A), \tag{13}$$

$$S = \text{Softmax}(\text{MPNN}_{\text{c}}(X_a, E_a, A)), \tag{14}$$

where MPNN_{e} and MPNN_{c} are two-layer message-passing networks that learn to generate node embeddings ($Z \in \mathbb{R}^{n\times d}$) and a clustering matrix ($S \in \mathbb{R}^{n\times C}$), respectively; C denotes the number of substructures in molecules. A matrix of the pooled embeddings $X_s \in \mathbb{R}^{C\times d}$ for the substructures is computed:

$$X_s = S^T Z. \tag{15}$$

This learning-to-cluster module is placed after the atom-level encoder. Intuitively, atom nodes updated with both local and global information should be classified into accurate substructures.

C. Substructure-level Encoder

Given a set of substructures V_s with a matrix of embeddings $X_s \in \mathbb{R}^{C\times d}$, we forward X_s to L conventional Transformer encoder layers[19] to capture their pairwise semantics:

$$H^{(l)}_1 = \text{LayerNorm}(\text{SA}_l(H^{(l-1)}) + H^{(l-1)}), \tag{16}$$

$$H^{(l)} = \text{LayerNorm}(\text{FFN}_l(H^{(l)}_1) + H^{(l)}_1), \tag{17}$$

where SA refers to (multi-head) self-attention described in Eq. (6), and $H^{(0)}$ is equal to X_s. Additionally, we add a long-range skip connection to alleviate gradient vanishing as:

$$H_s = \text{FFN}(\text{concat}(H^{(0)}, H^{(\text{L})})), \tag{18}$$

where $H_s \in \mathbb{R}^{C\times d}$ is the output indicating the representations for the substructures. Finally, we aggregate all C vectors $h_s \in H_s$ to result in a unique representation $z \in \mathbb{R}^d$ for the

molecules (refer to Section III C), before feeding it to a feed-forward network to compute the final output $\hat{y} \in \mathbb{R}_c^n$ for property prediction:

$$z = \mathbf{R}(\{h_s\}_{s=1}^{C}), \tag{19}$$
$$\hat{y} = \text{FFN}(z). \tag{20}$$

a. Training Objective We train MGT by minimizing the objective L that is defined as:

$$L = L_1 + \lambda_1 L_{LP} + \lambda_2 L_E, \tag{21}$$

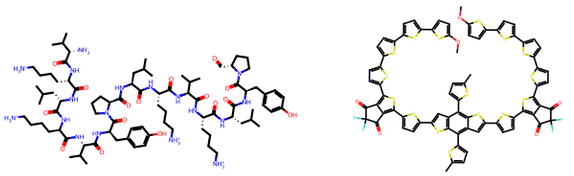
where $L_1 \triangleq \ell(\hat{y}, y)$ denotes the loss function between predicted values and ground truths (e.g., cross-entropy or mean-squared error), $L_{LP} \triangleq \|A - SS^T\|_F$ indicates auxiliary link prediction loss ($\|\cdot\|_F$ denotes the Frobenius norm), and $L_E \triangleq \frac{1}{n}\sum_{i=1}^{n} \text{H}(S_i)$ denotes the entropy regularization of the cluster assignment, i.e. each atom should be assigned into a unique cluster. Additionally, λ_1 and λ_2 are hyperarameters.

VI. FAST COMPUTATION ON LARGE GRAPHS

Although Transformers with graph spectral encoding schemes offer remarkable performance in capturing long-range interactions on molecular graphs, these approaches are not scalable when the graph sizes become larger. Computing eigenvalues on large graphs remains a challenging problem as it requires $O(n^3)$ time complexity where n is the number of nodes. Furthermore, Transformers-based models (see Section III D) that work on fully-connection graphs also have scalability drawbacks as they exhibit quadratic computations[64].

Although equivariant encoding schemes are theoretically expensive when dealing with fourth-order tensors (see Eq. 9), we do not need to store all elements of these tensors by leveraging their sparsity (i.e. ignoring all zero entries in the adjacency matrix). This can reduce the space complexity to less than $O(n^4)$ in the case of fourth-order tensors, where n is the number of nodes. Regarding the time complexity, it is proportional to $O(mn^2)$, where m denotes the number of edges. In fact, molecules are generally tree-like structures; hence, $m = O(n)$. Therefore, the theoretical time complexity of general equivariant encoding is $O(n^3)$[56]. However, in practice, we can further improve this complexity extensively by pre-computation for some special cases of contractions like in Figure 3.

In this section, we present several modifications to our proposed methods that can scale to graphs of hundreds of atoms, such as protein-ligand complexes. To accelerate positional encoding on a large graph, we partition it into multiple separate subgraphs and compute graph wavelets on each of them. We reduce tensor contractions to diagonalization, row sum, and column sum. Furthermore, by design, the self-attention layer in Equation 12 of the atom-level encoder in Section V A can be eliminated without affecting the entire pipeline, and the atom nodes only exchange the messages within their neighborhoods. This, as a result, prevents quadratic computations of the self-attention mechanism when the graph becomes exceedingly large. Albeit only computing local interactions in the atom-level encoder, MGT can take into account interactions between distant atoms at the substructure-level resolution as embeddings of the atoms are aggregated into their corresponding substructures.



(a) (b)

FIG. 5: Examples of two macromolecules. a) An example of a peptide that consists of many functional groups. b) An example of a polymer that consists of many repeating units

VII. EXPERIMENTS

We empirically validate our proposed approach in two types of macromolecules including peptides and polymers. Figure 5 illustrates two examples of macromolecules in the datasets. Our PyTorch implementation is publicly available at `https://github.com/HySonLab/Multires-Graph-Transformer`.

A. Polymer Property Prediction

In modern quantum chemistry, density functional theory (DFT) is the most widely used and successful approach for computing the electronic structure of matter. Even though DFT is able to calculate many properties of molecular systems with high accuracy, the computational cost is significantly expensive, especially for macromolecules with hundreds or thousands of atoms. Polymers are long chains of repetitive substructures known as repeating units. They, as a result, are also hierarchically structured and contain various types of long-range dependencies among the atoms. Since polymers have a wide range of applications in daily life, it is essential to understand their molecular properties. In this section, we demonstrate the efficacy of our proposed model, MGT, along with atomic positional encoding in estimating DFT calculation on polymers.

a. Experimental Setup We use a polymer dataset proposed in[65]. Each polymer is associated with three types of density functional theory (DFT)[66] properties including the first excitation energy of the monomer calculated with time-dependent DFT (GAP), the energy of the highest occupied molecular orbital for the monomer (HOMO), and the lowest unoccupied molecular orbital of the monomer (LUMO). The dataset is split into train/validation/test subsets with a ratio of 8:1:1, respectively. For training, we normalize all learning targets with a mean of 0 and a standard deviation of 1.

b. Baselines and Implementation Details As there are no existing baselines on this dataset, we perform experiments with

Model	No. Params	Property		
		GAP	HOMO	LUMO
DFT error		1.2	2.0	2.6
Chemical accuracy		0.043	0.043	0.043
GCN	527k	0.1094 ± 0.0020	0.0648 ± 0.0005	0.0864 ± 0.0014
GCN + Virtual Node	557k	0.0589 ± 0.0004	0.0458 ± 0.0007	0.0482 ± 0.0010
GINE	527k	0.1018 ± 0.0026	0.0749 ± 0.0042	0.0764 ± 0.0028
GINE + Virtual Node	557k	0.0870 ± 0.0040	0.0565 ± 0.0050	0.0524 ± 0.0010
GPS	600k	0.0467 ± 0.0010	0.0322 ± 0.0020	0.0385 ± 0.0006
Transformer + LapPE	700k	0.2949 ± 0.0481	0.1200 ± 0.0206	0.1547 ± 0.0127
MGT + LapPE (ours)	499k	$\mathbf{0.0378 \pm 0.0004}$	$\mathbf{0.0270 \pm 0.0010}$	0.0300 ± 0.0006
MGT + RWPE (ours)	499k	0.0384 ± 0.0015	0.0274 ± 0.0005	$\mathbf{0.0290 \pm 0.0007}$
MGT + WavePE (ours)	499k	0.0387 ± 0.0011	0.0283 ± 0.0004	0.0290 ± 0.0010

TABLE I: Experimental results on the polymer property prediction task. All the methods are trained in four different random seeds and evaluated by MAE ↓. Our methods are able to attain better performance across three DFT properties of polymers while having less number of parameters. All the properties are measured in eV.

Model	No.Params	Peptides-struct	Peptides-func
		MAE ↓	AP ↑
GCN	508k	0.3496 ± 0.0013	0.5930 ± 0.0023
GINE	476k	0.3547 ± 0.0045	0.5498 ± 0.0079
GatedGCN	509k	0.3420 ± 0.0013	0.5864 ± 0.0077
GatedGCN + RWPE	506k	0.3357 ± 0.0006	0.6069 ± 0.0035
Transformer + LapPE	488k	0.2529 ± 0.0016	0.6326 ± 0.0126
GPS	—	0.6535 ± 0.0041	0.2500 ± 0.0005
SAN + LapPE	493k	0.2683 ± 0.0043	0.6384 ± 0.0121
SAN + RWPE	500k	0.2545 ± 0.0012	0.6562 ± 0.0075
MGT + LapPE (ours)	499k	0.2488 ± 0.0014	0.6728 ± 0.0152
MGT + RWPE (ours)	499k	0.2496 ± 0.0009	0.6709 ± 0.0083
MGT + WavePE (ours)	499k	$\mathbf{0.2453 \pm 0.0025}$	$\mathbf{0.6817 \pm 0.0064}$

TABLE II: Results on peptides property prediction.

four different models for comparisons. For local GNNs, we use GCN[10] and GINE[13,67] augmented with virtual nodes as the baselines. The implementation of local GNN models is taken from `https://github.com/snap-stanford/ogb/tree/master/ogb`. Moreover, we use standard Transformer[19] with Laplacian positional encoding[28] and GPS[25] as the baselines for Transformer-based architectures. The implementation of MGT is similar to the Peptide tasks.

Table III shows the implementation of the baselines we used in the polymer experiments. All the models are designed to have approximately 500 to 700k learnable parameters. For fair comparisons, all the models are trained in 50 epochs with a learning rate of 0.001 and batch size of 128.

c. Results As shown in Table I, our MGT models achieve the lowest MAE scores across three properties. In addition, WavePE can attain comparable results with LapPE and RWPE for this task. We observe that the vanilla Transformer has the poorest performance. This demonstrates that computing global information without the awareness of locality is not sufficient for macromolecular modeling. As described in Section V, MGT is an extended version of GPS. In particular, a learning-to-cluster module and a substructure-level Transformer encoder are extensions to GPS. The better performance of MGT, as a result, indicates that our methodology in modeling hierarchical structures is appropriate and reasonable.

Model	No. Layer	Embed Dim	No. Params
GCN	5	156	527k
GCN + Virtual Node	5	156	557k
GINE	5	120	527k
GINE + Virtual Node	5	120	557k
GPS	3	120	600k
Transformer + LapPE	6	120	700k

TABLE III: The detailed settings of baselines for polymer property prediction

There are two important benchmark error levels: (1) "DFT error", the estimated average error of the DFT approximation to nature; and (2) "chemical accuracy", the target error that

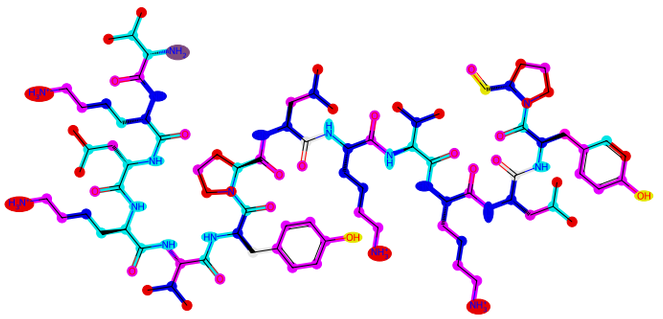


FIG. 6: The clustering result on a peptide. MGT can group the atoms of a long peptide into different substructure types. Specifically, the groups NH3 and OH are recognized even though the atoms are located distantly. Also, local rings or segments are also detected.

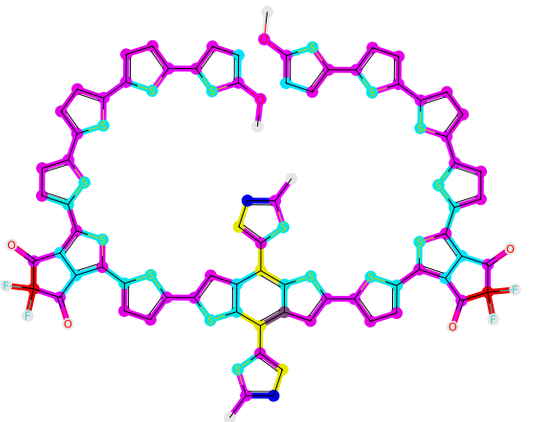


FIG. 7: The clustering result on a polymer. By learning to cluster and using a substructure-level Transformer encoder, MGT can model repetitive patterns existing in polymers. In this example, the model can recognize repeating units in a long-chain polymer or even symmetries.

has been established by the chemistry community. Estimates of DFT error and chemical accuracy are provided by Faber *et al.* [68]. Our model is the only one to achieve the chemical accuracy for all three molecular properties.

B. Peptides Property Prediction

Peptides are small chains of amino acids found in nature that serve a variety of critical biological roles[69]; however, they are far shorter than proteins. Because each amino acid is made up of several heavy atoms, a peptide's molecular graph is significantly greater than that of a small drug-like molecule. Since peptides are formed by sequences of amino acids, they are naturally hierarchical and long-range dependencies[64], i.e. a peptide should be ideally segmented into an exact set of amino acids. Therefore, we evaluate our method on peptide structures to demonstrate its superiority.

a. Experimental Setup We run experiments on two real-world datasets including (1) Peptides-struct and (2) Peptides-func of the Long-range Graph Benchmark[64]. The two datasets are multi-label graph classification problems and share the same peptide molecular graphs, but with different tasks. While the former consists of 10 classes based on peptides function, the latter is used to predict 11 aggregated 3D properties of peptides at the graph level. For a fair comparison, we follow the experimental and evaluation setting of[64] with the same train/test split ratio. We use mean absolute error (MAE) and average precision (AP) to evaluate the method's performance for Peptides-struct and Peptides-func, respectively.

b. Baselines and Implementation Details We compare our proposed approach with the baselines from[64]. The local message-passing network class involves GCN[10], GCNII[52], GINE[13,67], and GatedGCN[52]. For Transformer-based architectures, we compare our method with vanilla Transformer[19] with Laplacian PE[28,70] and SAN[23]. Since all baselines are limited to approximately 500k learnable parameters, we also restrict MGT to roughly the same number of parameters. Additionally, we use GatedGCN[52] for local message passing customized with the PEG technique to stabilize the positional features[71]. The implementation of GPS is adapted from `https://github.com/vijaydwivedi75/lrgb.git`. We experiment with each task in four different random seeds. We provide further implementation details of MGT for this task in Section VII G.

c. Results Table II shows that our proposed MGT + WavePE achieves the best performances in two peptide prediction tasks. In addition to WavePE, MGT + RWPE also attains the second-best performances. The superiority of WavePE to RWPE can be explained as mentioned in Section IV that WavePE is a generalized version of RWPE. In particular, our proposed MGT outperforms all the baselines in the Petides-func task by a large margin and decreases the MAE score to less than 0.25 in the Petides-struct task.

C. Protein-Ligand binding affinity prediction

Proteins are large and complex macromolecules that comprise one or more long chains of amino acid residues. Understanding the multiscale structure of proteins is important in estimating their fitness and functionality. In this experiment, we show the effectiveness of our model in capturing the long-range and hierarchical structures of proteins, that are larger than the peptides from Section VII B.

a. Experimental Setup Predicting the binding affinities between ligands and target pockets is an essential task in the field of drug discovery. In this section, we present our experiments conducted on the Ligand Binding Affinity

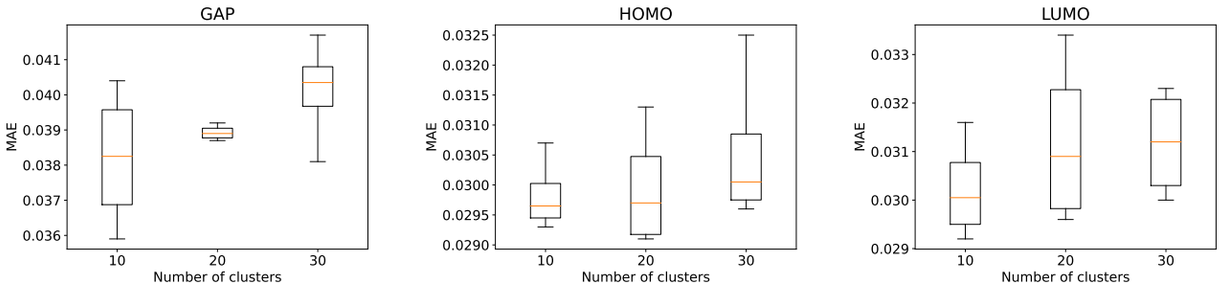


FIG. 8: Ablation studies on different numbers of clusters on polymer datasets

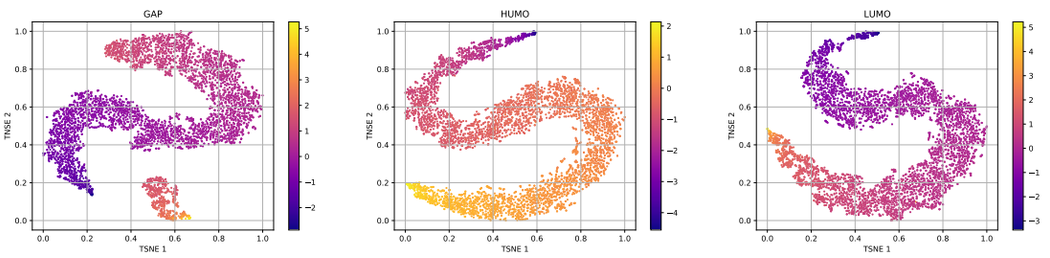
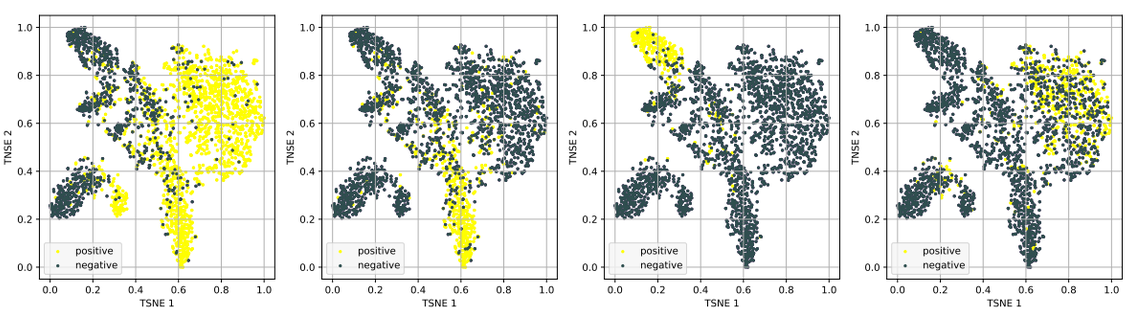
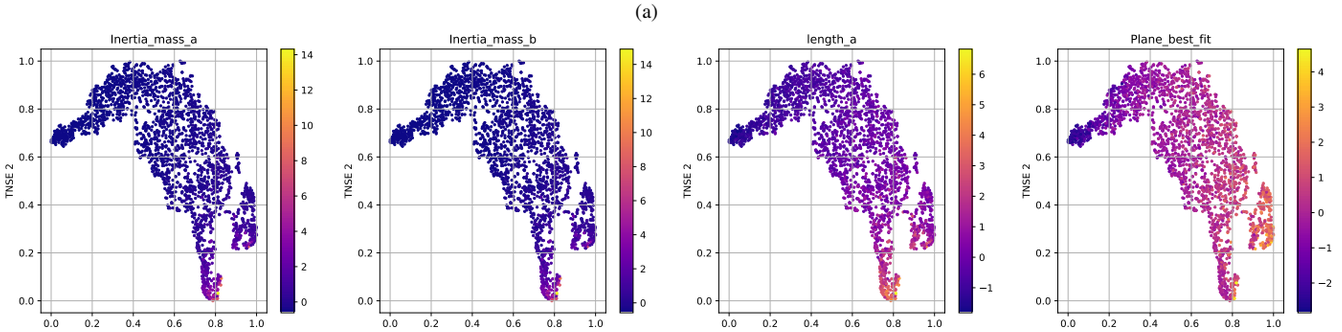


FIG. 9: t-SNE projection of representations of the test polymers in the dataset. We plot the figures of three properties, including GAP, HUMO, and LUMO. As the labels are continuous values, points are color-coded by spectrums wherein higher values correspond to warmer palettes.



(a)



(b)

FIG. 10: Visualization of low-dimensional spaces of peptides on two property prediction tasks: Peptides-func and Peptides-struct. All the vectors are normalized to range $[0, 1]$. a) t-SNE projection of peptides taken from the Peptides-func testing dataset. We take four random peptide functions, and each figure corresponds to one of the properties with positive (1) and negative (0) ground truths. b) Similarly, we plot the figures of four random peptide properties taken from the Peptides-struct testing dataset. The spectrums represent continuous ground truths, where lower values correspond to cooler colors.

Method	3D-CNN	GNN	ENN	GVP-GNN	MGT + WavePE (ours)
RMSE ↓	**1.416 ± 0.021**	1.570 ± 0.025	1.568 ± 0.012	1.594 ± 0.073	1.436 ± 0.066

TABLE IV: Experimental Results on LBA dataset

(LBA) dataset within the Atom3D benchmark[72]. The dataset comprises protein-ligand complexes curated from the PDBBind database[73,74] and their corresponding binding affinities. Moreover, the complexes are represented in terms of 3D point clouds of atoms with 3D coordinates and atomic features. Our models are trained to predict the value of $pK = -\log_{10}(K)$, where K represents the binding affinity expressed in Molar units. We follow the original train/test split settings established by Townshend *et al.* [72]. Finally, we report the root-mean-squared error (RMSE) as the primary metric to compare with other approaches.

b. Baselines and Implementation Details Following the description from Townshend *et al.* [72], we construct 2D graphs from molecular systems where each atom represents a node. Edges are determined between atoms that are distant less than 4.5 Å and their weights are defined as $w_{i,j} = \frac{1}{d_{i,j}}$. We compare the performance of our methods with baselines taken from [71] including:

- 3D-CNN: Three-dimensional Convolutional Neural Networks on the voxelized representation of the protein-ligand complex's point cloud,
- GNN: Graph Neural Networks on the atomic-level graph,
- ENN: Rotationally-Equivariant Neural Networks[75] on the 3D point cloud representation,
- GVP-GNN: Geometric Vector Perception (GVP)[71,76], a rotationally-invariant feature for protein, is incorporated into GNN.

Among these methods, except for GNN, the remaining models work on 3D geometries (e.g., voxels, point clouds, etc.) of protein-ligand complexes. For our MGT, we use the METIS algorithm[77] to partition the graphs into multiple separated subgraphs and compute relative Wavelet positional features for the nodes in each subgraph as described in Section V C. In addition, the implementation of MGT is almost similar to those used in the experiments on peptides and polymers in Section VII A and VII B, with the exception that the self-attention layer in Equation 11 is not utilized.

c. Results Table IV demonstrates the superior performance of our proposed methods compared to GNN, ENN, and GVP-GNN. Notably, the latter two approaches operate on protein-ligand complexes represented in 3D structures, whereas our method, MGT, operates on 2D graphs. Furthermore, our method is comparable with 3D-CNN, a 3D convolutional neural network working on voxels, which are computationally expensive when the complexes become larger.

D. Drug-like molecule property prediction

Method	No. Params	MAE ↓
GCN	505k	0.367 ± 0.011
GINE	510k	0.526 ± 0.051
GAT	531k	0.384 ± 0.007
PNA	387k	0.142 ± 0.010
MPNN	418k	0.145 ± 0.007
GatedGCN	505k	0.214 ± 0.006
SAN	509k	0.139 ± 0.006
Graphormer	489k	0.122 ± 0.006
GPS	-	0.070 ± 0.004
Spec-GN	503k	0.0698 ± 0.002
MGT + WavePE (ours)	499k	0.131 ± 0.003

TABLE V: Experimental results on the ZINC-12K dataset

Although MGT is intentionally designed for learning to represent hierarchical structures, we report its experimental results on the ZINC-12K dataset[3], which consists of small drug-like molecules, in this section. We train MGT to predict the solubility (LogP) of the molecules with up to 50 heavy atoms on a subset of the ZINC dataset. We follow the split of 10K/1K/1K for training/validation/testing proposed by Dwivedi *et al.* [70]. Baseline results include GCN[10], GINE[13,70], , GAT[11], Spec-GN[78], PNA[79], GatedGCN[52], GPS[25], MPNN[51], SAN[23], DGN[80], and Graphormer[39]. According to Table V, MGT + WavePE outperforms 7 out of 10 other baselines.

E. Ablation study

The ablation study, presented in Figure 8, is designed to explore the role of the number of clusters for our proposed multiscale learning method. In particular, we conduct experiments by varying the number of clusters C when learning on polymer datasets across three properties, i.e. GAP, HOMO, and LUMO. The implications suggest that increasing the number of clusters consistently leads to superior performances across all properties. This is due to the fact that macromolecules like polymers are constituted by repeating units, and increasing the number of clusters may break this inductive bias. Furthermore, the plots also demonstrate that using predefined numbers of clusters can lead to unstable performance across all tasks. As datasets grow larger, different molecules exhibit diverse substructures. Consequently, using a fixed number of clusters for all data samples may not be sufficient for effective generalization. This paves the way for future research to determine the specific number of clusters for each molecule adaptively in a data-driven manner.

F. Visualization

We use the t-SNE algorithm[81] to project the representations produced by MGT (with WavePE) of peptides and polymers of the test datasets into two-dimensional spaces for visualization. Also, we take the probabilistic clustering matrix $\mathbf{S}$ in Equation (14) to visualize the clustering results on the molecules. Specifically, we use the RDKit package (open-source cheminformatics: `https://www.rdkit.org`) to draw the molecules. Figures 10 and 9 show clear and smooth clustering patterns in low-dimensional spaces that indicate our proposed approaches are able to learn meaningful molecular representations for hierarchical structures such as peptides and polymers. Furthermore, according to Figures 6 and 7, our learning-to-cluster algorithm and multiresolution analysis can pick up functional groups (for proteins/peptides) and repeating units (for polymers) via back-propagation.

G. Implementation Details

Hyperparameters	Values
No. Epoch	200
Embedding Dimension	84
Batch size	128
Learning rate	0.001
Dropout	0.25
Attention Dropout	0.5
Diffusion Step (k)	[1, 2, 3, 4, 5]
No. Head	4
Activation	ReLU
Normalization	Batchnorm
No. Cluster	10
λ_1	0.001
λ_2	0.001

TABLE VI: The hyperparameters for MGT

In this section, we elaborate on the architecture and hyperparameters used to train and evaluate our MGT to achieve the above numerical results. Table VI shows details of the hyperparameters used for MGT in all the experiments. In particular, we use the atom and bond encoder modules provided by OGB[82] to attribute the molecular graph. To compute the wavelet tensors, we define a set of diffusion steps k (i.e. scales). Intuitively, the determination of these scales involves progressively expanding the number of hops around the central nodes, thereby effectively capturing and quantifying the extent to which information spreads throughout the graph. We use two GPS layers to compute the atom-level embeddings and two Transformer layers for calculating the substructure-level embeddings. For learning to cluster, we use a 2-layer message-passing network to compute $\mathbf{Z}$ and $\mathbf{S}$ mentioned in Eq. (13) (14) as follows:

$$\mathbf{Z}_a^1, \mathbf{E}_a^1 = \text{GatedGCN}^1(\mathbf{X}_a, \mathbf{E}_a, \mathbf{A}), \tag{22}$$
$$\mathbf{Z}_a^1 = \text{Batchnorm}(\text{ReLU}(\mathbf{Z}_a^1)), \tag{23}$$
$$\mathbf{Z}_a^2, \mathbf{E}_a^2 = \text{GatedGCN}^2(\mathbf{Z}_a^1, \mathbf{E}_a^1, \mathbf{A}), \tag{24}$$
$$\mathbf{Z}_a^2 = \text{Batchnorm}(\text{ReLU}(\mathbf{Z}_a^2)), \tag{25}$$
$$\mathbf{Z} = \text{concat}(\mathbf{Z}_a^1, \mathbf{Z}_a^2), \tag{26}$$
$$\mathbf{Z} = \text{FFN}(\mathbf{Z}), \tag{27}$$

in which $\mathbf{S}$ is computed similarly with an auxiliary Softmax operation on the output to produce a probabilistic clustering matrix.

VIII. CONCLUSION

In this paper, we introduce a novel architecture, Multiresolution Graph Transformer (MGT), that is able to learn and capture the molecular structure of macromolecules at multiple levels of resolution. We utilize the popular Transformer architecture to model the long-range atomic interaction. Our proposed model employs a learning-to-cluster algorithm that is trainable via back-propagation in order to construct the hierarchy of coarse-graining graphs (i.e. multiresolution) while detecting important functional groups. Furthermore, to empower MGT, we propose a new atomic positional encoding named WavePE based on multiresolution analysis and wavelet theory. We have shown competitive experimental results on three macromolecule datasets of polymers, peptides, and protein-ligand complexes, and one small molecule dataset of drug-like compounds. Noticeably, our model achieves the chemical accuracy in approximating the density functional theory (DFT) calculation and outperforms other state-of-the-art graph learning methods in the polymers dataset. We have released our software and data publicly. We believe our model and implementation will certainly advance the field of DFT approximation for large-scale molecular structures and allow several downstream applications in drug discovery and materials science.

[1] L. Ruddigkeit, R. Deursen, L. Blum, and J.-L. Reymond, "Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17," Journal of chemical information and modeling **52** (2012), 10.1021/ci300415d.
[2] R. Ramakrishnan, P. Dral, M. Rupp, and A. von Lilienfeld, "Quantum chemistry structures and properties of 134 kilo molecules," Scientific Data **1** (2014), 10.1038/sdata.2014.22.
[3] T. Sterling and J. Irwin, "Zinc 15 - ligand discovery for everyone," Journal of chemical information and modeling **55** (2015), 10.1021/acs.jcim.5b00559.
[4] M. Jerry, "Advanced organic chemistry: reactions, mechanisms and structure," (1992).
[5] F. Schmid, "Understanding and modeling polymers: The challenge of multiple scales," ACS Polymers Au **0**, null (2022), https://doi.org/10.1021/acspolymersau.2c00049.
[6] G. Anand, S. Ghosh, L. Zhang, A. Anupam, C. L. Freeman, C. Ortner, M. Eisenbach, and J. R. Kermode, "Exploiting machine learning in multiscale modelling of materials," Journal of The Institution of Engineers (India): Series D (2022), 10.1007/s40033-022-00424-z.
[7] C. Gaul and S. Cuesta-Lopez, "Machine learning for screening large organic molecules," arXiv preprint arXiv:2211.15415 (2022).

[8]P. N. Depta, M. Dosta, W. Wenzel, M. Kozlowska, and S. Heinrich, "Hierarchical coarse-grained strategy for macromolecular self-assembly: Application to hepatitis b virus-like particles," International Journal of Molecular Sciences **23** (2022), 10.3390/ijms232314699.

[9]J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 70, edited by D. Precup and Y. W. Teh (PMLR, 2017) pp. 1263–1272.

[10]T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv preprint arXiv:1609.02907 (2016).

[11]P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," International Conference on Learning Representations (2018).

[12]G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Veličković, "Principal neighbourhood aggregation for graph nets," in *Advances in Neural Information Processing Systems*, Vol. 33, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (Curran Associates, Inc., 2020) pp. 13260–13271.

[13]K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019* (OpenReview.net, 2019).

[14]C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19 (AAAI Press, 2019).

[15]D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," Proceedings of the AAAI Conference on Artificial Intelligence **34**, 3438–3445 (2020).

[16]Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18 (AAAI Press, 2018).

[17]K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," in *International Conference on Learning Representations* (2020).

[18]U. Alon and E. Yahav, "On the bottleneck of graph neural networks and its practical implications," in *International Conference on Learning Representations* (2021).

[19]A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17 (Curran Associates Inc., Red Hook, NY, USA, 2017) p. 6000–6010.

[20]J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (Association for Computational Linguistics, Minneapolis, Minnesota, 2019) pp. 4171–4186.

[21]A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations* (2021).

[22]Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2021).

[23]D. Kreuzer, D. Beaini, W. Hamilton, V. Létourneau, and P. Tossou, "Rethinking graph transformers with spectral attention," Advances in Neural Information Processing Systems **34**, 21618–21629 (2021).

[24]V. P. Dwivedi and X. Bresson, "A generalization of transformer networks to graphs," CoRR **abs/2012.09699** (2020), 2012.09699.

[25]L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a General, Powerful, Scalable Graph Transformer," arXiv:2205.12454 (2022).

[26]C. Cai, T. S. Hy, R. Yu, and Y. Wang, "On the connection between mpnn and graph transformer," arXiv preprint arXiv:2301.11956 (2023).

[27]J. Kim, D. T. Nguyen, S. Min, S. Cho, M. Lee, H. Lee, and S. Hong, "Pure transformers are powerful graph learners," in *Advances in Neural Information Processing Systems*, edited by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho (2022).

[28]V. P. Dwivedi and X. Bresson, "A generalization of transformer networks to graphs," CoRR **abs/2012.09699** (2020), 2012.09699.

[29]V. P. Dwivedi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, "Graph neural networks with learnable structural and positional representations," in *International Conference on Learning Representations* (2022).

[30]J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 97, edited by K. Chaudhuri and R. Salakhutdinov (PMLR, 2019) pp. 7134–7143.

[31]P. Li, Y. Wang, H. Wang, and J. Leskovec, "Distance encoding: Design provably more powerful neural networks for graph representation learning," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20 (Curran Associates Inc., Red Hook, NY, USA, 2020).

[32]D. Lim, J. Robinson, L. Zhao, T. Smidt, S. Sra, H. Maron, and S. Jegelka, "Sign and basis invariant networks for spectral graph representation learning," arXiv preprint arXiv:2202.13013 (2022).

[33]D. Chen, L. O'Bray, and K. Borgwardt, "Structure-aware transformer for graph representation learning," in *Proceedings of the 39th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research (2022).

[34]Y. Fang, H. Yang, X. Zhuang, X. Shao, X. Fan, and H. Chen, "Knowledge-aware contrastive molecular graph learning," CoRR **abs/2103.13047** (2021), 2103.13047.

[35]Y. Rong, Y. Bian, T. Xu, W. Xie, Y. Wei, W. Huang, and J. Huang, "Self-supervised graph transformer on large-scale molecular data," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20 (Curran Associates Inc., Red Hook, NY, USA, 2020).

[36]F. Chen, J. Park, and J. Park, "A hypergraph convolutional neural network for molecular properties prediction using functional group," CoRR **abs/2106.01028** (2021), 2106.01028.

[37]K. Maziarz, H. R. Jackson-Flux, P. Cameron, F. Sirockin, N. Schneider, N. Stiefl, M. Segler, and M. Brockschmidt, "Learning to extend molecular scaffolds with structural motifs," in *International Conference on Learning Representations* (2022).

[38]W. Jin, R. Barzilay, and T. Jaakkola, "Hierarchical generation of molecular graphs using structural motifs," in *Proceedings of the 37th International Conference on Machine Learning*, ICML'20 (JMLR.org, 2020).

[39]C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu, "Do transformers really perform badly for graph representation?" in *Advances in Neural Information Processing Systems*, edited by A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan (2021).

[40]S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," in *Advances in Neural Information Processing Systems*, Vol. 32, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019).

[41]S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," IEEE Transactions on Pattern Analysis and Machine Intelligence **11**, 674–693 (1989).

[42]S. Mallat, *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*, 3rd ed. (Academic Press, Inc., USA, 2008).

[43]A. Haar, "Zur theorie der orthogonalen funktionensysteme," Mathematische Annalen **69**, 331–371 (1910).

[44]I. Daubechies, "Orthonormal bases of compactly supported wavelets," Communications on Pure and Applied Mathematics **41**, 909–996 (1988).

[45]D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," Applied and Computational Harmonic Analysis **30**, 129–150 (2011).

[46]R. R. Coifman and M. Maggioni, "Diffusion wavelets," Applied and Computational Harmonic Analysis **21**, 53–94 (2006), special Issue: Diffusion Maps and Wavelets.

[47]R. M. Rustamov and L. Guibas, "Wavelets on graphs via deep learning," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13 (Curran Associates Inc., Red Hook, NY, USA, 2013) p. 998–1006.

[48]B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, "Graph wavelet neural network," in *International Conference on Learning Representations* (2019).

[49]T. S. Hy and R. Kondor, "Multiresolution matrix factorization and wavelet networks on graphs," in *Proceedings of Topological, Algebraic, and Geometric Learning Workshops 2022*, Proceedings of Machine Learning Research, Vol. 196, edited by A. Cloninger, T. Doster, T. Emerson, M. Kaul, I. Ktena, H. Kvinge, N. Miolane, B. Rice, S. Tymochko, and G. Wolf (PMLR, 2022) pp. 172–182.

[50]R. Kondor, N. Teneva, and V. Garg, "Multiresolution matrix factorization," in *Proceedings of the 31st International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 32, edited by E. P. Xing and T. Jebara (PMLR, Bejing, China, 2014) pp. 1620–1628.

[51]J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17 (JMLR.org, 2017) p. 1263–1272.

[52]X. Bresson and T. Laurent, "Residual gated graph convnets," CoRR **abs/1711.07553** (2017), 1711.07553.

[53]M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," Neural Computation **15**, 1373–1396 (2003).

[54]C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18 (Association for Computing Machinery, New York, NY, USA, 2018) p. 1320–1329.

[55]B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, "Graph wavelet neural network," in *International Conference on Learning Representations* (2019).

[56]T. S. Hy, S. Trivedi, H. Pan, B. M. Anderson, , and R. Kondor, "Predicting molecular properties with covariant compositional networks," The Journal of Chemical Physics **148** (2018).

[57]R. Kondor, T. S. Hy, H. Pan, S. Trivedi, and B. M. Anderson, "Covariant compositional networks for learning graphs," Proc. ICLR Workshop (2018).

[58]H. Maron, H. Ben-Hamu, N. Shamir, and Y. Lipman, "Invariant and equivariant graph networks," in *International Conference on Learning Representations* (2019).

[59]N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," Journal of Machine Learning Research **15**, 1929–1958 (2014).

[60]S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," CoRR **abs/1502.03167** (2015), 1502.03167.

[61]L. J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," CoRR **abs/1607.06450** (2016), 1607.06450.

[62]R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18 (Curran Associates Inc., Red Hook, NY, USA, 2018) p. 4805–4815.

[63]T. S. Hy and R. Kondor, "Multiresolution equivariant graph variational autoencoder," Machine Learning: Science and Technology **4**, 015031 (2023).

[64]V. P. Dwivedi, L. Rampasek, M. Galkin, A. Parviz, G. Wolf, A. T. Luu, and D. Beaini, "Long range graph benchmark," in *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track* (2022).

[65]P. C. St. John, C. Phillips, T. W. Kemper, A. N. Wilson, Y. Guan, M. F. Crowley, M. R. Nimlos, and R. E. Larsen, "Message-passing neural networks for high-throughput polymer screening," The Journal of chemical physics **150**, 234111 (2019).

[66]P. Hohenberg and W. Kohn, "Inhomogeneous electron gas," Phys. Rev. **136**, B864–B871 (1964).

[67]W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," in *International Conference on Learning Representations* (2020).

[68]F. A. Faber, L. Hutchison, B. Huang, J. Gilmer, S. S. Schoenholz, G. E. Dahl, O. Vinyals, S. Kearnes, P. F. Riley, and O. A. von Lilienfeld, "Machine learning prediction errors better than dft accuracy," arXiv preprint arXiv:1702.05532 (2017).

[69]S. Singh, K. Chaudhary, S. K. Dhanda, S. Bhalla, S. S. Usmani, A. Gautam, A. Tuknait, P. Agrawal, D. Mathur, and G. P. Raghava, "Satpdb: a database of structurally annotated therapeutic peptides," Nucleic acids research **44**, D1119–D1126 (2016).

[70]V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," CoRR **abs/2003.00982** (2020), 2003.00982.

[71]H. Wang, H. Yin, M. Zhang, and P. Li, "Equivariant and stable positional encoding for more powerful graph neural networks," in *International Conference on Learning Representations* (2022).

[72]R. J. Townshend, M. Vögele, P. Suriana, A. Derry, A. Powers, Y. Laloudakis, S. Balachandar, B. Jing, B. Anderson, S. Eismann, *et al.*, "Atom3d: Tasks on molecules in three dimensions," arXiv preprint arXiv:2012.04035 (2020).

[73]R. Wang, X. Fang, Y. Lu, and S. Wang, "The pdbbind database: Collection of binding affinities for protein- ligand complexes with known three-dimensional structures," Journal of medicinal chemistry **47**, 2977–2980 (2004).

[74]Z. Liu, Y. Li, L. Han, J. Li, J. Liu, Z. Zhao, W. Nie, Y. Liu, and R. Wang, "Pdb-wide collection of binding data: current status of the pdbbind database," Bioinformatics **31**, 405–412 (2015).

[75]B. Anderson, T.-S. Hy, and R. Kondor, "Cormorant: Covariant molecular neural networks," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (Curran Associates Inc., Red Hook, NY, USA, 2019).

[76]B. Jing, S. Eismann, P. Suriana, R. J. L. Townshend, and R. Dror, "Learning from protein structure with geometric vector perceptrons," in *International Conference on Learning Representations* (2021).

[77]G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," SIAM Journal on Scientific Computing **20**, 359–392 (1998), https://doi.org/10.1137/S1064827595287997.

[78]M. Yang, Y. Shen, R. Li, H. Qi, Q. Zhang, and B. Yin, "A new perspective on the effects of spectrum in graph neural networks," in *Proceedings of the 39th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 162, edited by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato (PMLR, 2022) pp. 25261–25279.

[79]G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Velickovic, "Principal neighbourhood aggregation for graph nets," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20 (Curran Associates Inc., Red Hook, NY, USA, 2020).

[80]G. Li, C. Xiong, A. K. Thabet, and B. Ghanem, "Deepergcn: All you need to train deeper gcns," CoRR **abs/2006.07739** (2020), 2006.07739.

[81]L. van der Maaten and G. Hinton, "Visualizing data using t-sne," Journal of Machine Learning Research **9**, 2579–2605 (2008).

[82]W. Hu, M. Fey, H. Ren, M. Nakata, Y. Dong, and J. Leskovec, "Ogb-lsc: A large-scale challenge for machine learning on graphs," arXiv preprint arXiv:2103.09430 (2021).