
Towards Multiscale Graph-based Protein Learning with Geometric Secondary Structural Motifs

Shih-Hsin Wang¹, Yuhao Huang¹, Taos Transue¹, Justin Baker²,
Jonathan Forstater³, Thomas Strohmmer³ & Bao Wang^{1*}

¹Department of Mathematics and Scientific Computing and Imaging (SCI) Institute
University of Utah, Salt Lake City, UT 84102, USA

²Department of Mathematics, UCLA, Los Angeles, CA 90095, USA

³Department of Mathematics, UC Davis, Davis, CA 95616, USA

Abstract

Graph neural networks (GNNs) have emerged as powerful tools for learning protein structures by capturing spatial relationships at the residue level. However, existing GNN-based methods often face challenges in learning multiscale representations and modeling long-range dependencies efficiently. In this work, we propose an efficient multiscale graph-based learning framework tailored to proteins. Our proposed framework contains two crucial components: (1) It constructs a hierarchical graph representation comprising a collection of fine-grained subgraphs, each corresponding to a secondary structure motif (e.g., α -helices, β -strands, loops), and a single coarse-grained graph that connects these motifs based on their spatial arrangement and relative orientation. (2) It employs two GNNs for feature learning: the first operates within individual secondary motifs to capture local interactions, and the second models higher-level structural relationships across motifs. Our modular framework allows a flexible choice of GNN in each stage. Theoretically, we show that our hierarchical framework preserves the desired maximal expressiveness, ensuring no loss of critical structural information. Empirically, we demonstrate that integrating baseline GNNs into our multiscale framework remarkably improves prediction accuracy and reduces computational cost across various benchmarks.

1 Introduction

Machine learning (ML) has transformed computational protein modeling over the past decade [20, 42]. The 2024 Nobel Prize in Chemistry, awarded for groundbreaking contributions to computational protein design, highlights the transformative advancements [1]. Among ML techniques, graph-based methods excel at learning to encode complex chemical interactions in three-dimensional space, effectively handling spatial information. For instance, prior works (cf. [14, 17]) represent atoms as nodes and leverage edges to capture their interactions, such as chemical bonds and hydrogen bonds. This edge-based encoding enables graph neural networks (GNNs) to effectively model critical chemical interactions, which are essential for understanding protein structure-function relationships.

However, representing proteins at the atomic level incurs significant computational costs due to the sheer size and

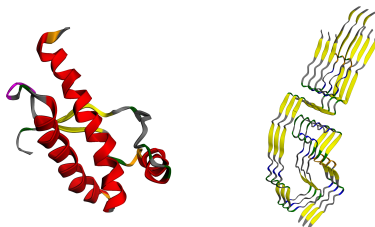


Figure 1: An example of two prion proteins with identical primary structures but distinct secondary structures. The normal form, hamster PrP^C (left), contains α -helical structures (marked in red). In contrast, its misfolded counterpart, PrP^{Sc}, on the right, lacks these helices and adopts a β -sheet-rich structure (marked in yellow). This structural change leads to abnormal aggregation, ultimately resulting in fatal consequences.

*Correspond to wangbaonj@gmail.com

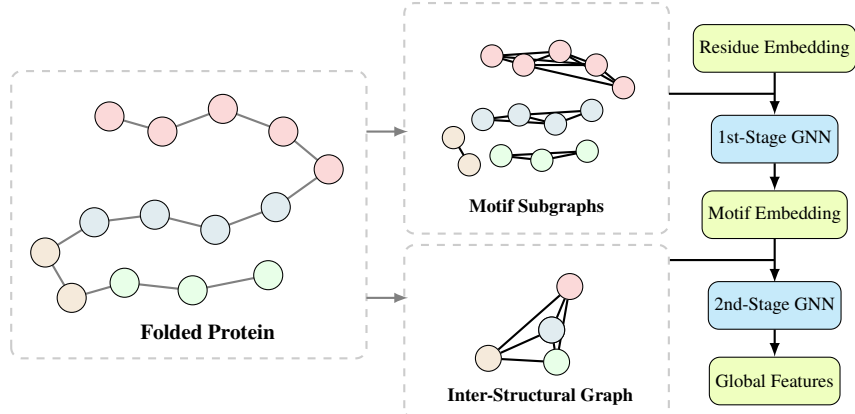


Figure 2: Overview of the proposed multiscale graph-based framework. We first construct a hierarchical graph representation that includes: (1) fine-grained motif subgraphs, where residues within each secondary structure motif (e.g., α -helices, β -strands, loops) are treated as nodes, and (2) a coarse-grained structural graph, where each motif is abstracted as a single node. The first GNN operates independently on each motif subgraph to learn local embeddings. These learned motif-level features are then used to construct the coarse-grained graph, on which a second GNN performs message passing to model higher-level structure and generate the final prediction.

complexity of the resulting graphs. To address this, recent methods have shifted towards residue-level representations, where each residue serves as a single node in the graph (cf. [18, 48, 37, 41]). This coarse-graining not only reduces the graph size but also aligns naturally with the primary structure of proteins, where residues are the fundamental building blocks that determine protein properties and overall structure. However, existing residue-level approaches face challenges in capturing critical multiscale features. In particular, secondary structures—such as α -helices and β -sheets—are formed by groups of residues and play a fundamental role in protein folding. Ignoring these higher-level structures can hinder the model’s capacity to distinguish between biologically distinct protein states that share identical primary sequences. An illustrative example is the prion protein [5]. The normal cellular form of the prion protein, PrP^{C} , is typically found on the surface of healthy neurons. However, it can misfold into the pathogenic form, PrP^{Sc} , without any change in its primary structure. The key difference lies in the spatial arrangement of residues, which are reorganized into different types of secondary structures, resulting in a markedly altered folding pattern compared to the normal form (see Section 3 for details). This misfolded form aggregates abnormally due to its β -sheet-rich structures and induces pathogenic effects, ultimately leading to fatal neurodegenerative diseases. Figure 1 shows hamster PrP^{C} (PDB: 1B10) and its misfolded form PrP^{Sc} (PDB ID: 7LNA) [23, 24].

Some multiscale methods have been proposed to capture hierarchical features of protein structures. For example, recent approaches use distance-thresholded graphs with large radial cutoffs, combined with surface modeling, to encode multiscale information [33, 47]. While large-cutoff, distance-thresholded graphs can capture broader structural context, they often introduce significant computational overhead—both in runtime and memory footprint (see Section 5 for empirical evidence)—and may overlook biologically meaningful relationships identified by domain experts. These limitations underscore the *urgent need for a scalable framework that integrates domain knowledge to model intricate protein interactions efficiently while preserving expressive power.*

1.1 Our Contributions

In this paper, we propose a new multiscale GNN framework—depicted in Fig. 2 with detailed discussion in Sections 3 and 4—for learning proteins. Our key contributions are:

- We introduce a hierarchical, sparse, and geometry-aware graph representation of proteins by combining domain-expert algorithms to segment sequences into secondary structure motifs and constructing a multi-scale graph hierarchy: a collection of fine-grained graphs that capture residue-level interactions within each secondary structure unit, and a single coarse-grained graph that models the spatial arrangement and relative orientation among these units. This representation preserves geometric fidelity while providing a provable sparsity bound on the total number of edges, which is critical for scalability and efficiency. See Section 3 for details.
- We develop a two-stage framework—leveraging two off-the-shelf GNNs operating in tandem to learn multiscale protein features based on the proposed hierarchical graph representation. The-

oretically, we characterize the maximal expressiveness of our framework, showing its ability to maintain spatial fidelity during message passing across different levels. See Section 4 for details.

- Empirically, we demonstrate that our multiscale framework enables existing GNN architectures to simultaneously improve prediction accuracy and computational efficiency (in both runtime and memory footprint) across benchmark tasks. See details in Section 5.

1.2 Related Works

Protein Representation Learning. A variety of deep learning approaches have been developed to model protein structures and functions by learning effective representations. Some methods leverage the sequential nature of proteins and employ convolutional neural networks (CNNs) [15] or large language models (LLMs) [45, 36] to learn directly from the amino acid sequence. However, since protein function is closely tied to its 3D structure, many recent efforts have shifted toward structure-based approaches. These methods represent proteins as graphs and use GNNs to capture spatial relationships. Notable examples of methods that learn geometric and symmetry-aware representations include [18, 17, 48, 37, 26, 41]. In addition, hybrid models such as DeepFRI [12] combine GNNs with sequence-level features extracted from pretrained protein language models, while ProtGO [16] integrates GNNs with a Gene Ontology encoder, achieving strong results with large-scale architectures. In contrast, our work focuses on 3D structure-based modeling and introduces an efficient, theoretically grounded hierarchical design that learns protein representation with substantially smaller model sizes.

Multiscale Graph-Based Models for Protein Representation Learning. Several recent efforts [14, 33, 47, 31] have explored multiscale GNN to better capture both local and global structural patterns. The methods in [33, 47] construct large-radius radial graphs combined with surface modeling, while [31] learns residue-level clustering for hierarchical representations. However, such approaches often incur high computational costs or rely on data-driven clustering without explicit biological grounding. In contrast, [14] incorporates domain principles by applying hierarchical pooling based on different types of residue interactions to extract multiscale features. However, their final pooling stages simplify the backbone chain by clustering every two consecutive residues along the sequence, without explicitly leveraging secondary structure motifs. As a result, capturing long-range dependencies (LRDs) requires repeated pooling over many layers, leading to high computational costs. Our work shares the multiscale motivation but differs by introducing a biologically grounded hierarchical graph construction inspired by prior molecular motif-based approaches [46, 43]. Specifically, we use secondary structures as high-level motifs to build a two-level graph that captures both local geometry and long-range dependencies. This design offers provable sparsity and expressiveness guarantees while maintaining high computational efficiency.

1.3 Organization

We organize this paper as follows: We recap on message-passing GNNs and the standard framework to analyze their expressiveness power, together with other necessary background materials in Section 2. We present our new hierarchical graph representations for proteins and two-stage GNN architectures in Section 3 and Section 4, respectively. We numerically validate the accuracy and efficiency of our proposed approach in Section 5. Technical proofs and additional experimental details are provided in the appendix.

2 Background

In this section, we provide background materials on point clouds, geometric graphs, local frames, and message-passing GNNs and their expressiveness characterizations.

Point Clouds and Geometric Graphs. A 3D point cloud is a collection of points in \mathbb{R}^3 , i.e., $\{\mathbf{x}_i\} \subset \mathbb{R}^3$. Each point may have a *feature vector* $\mathbf{f}_i \in \mathbb{R}^d$, capturing additional attributes beyond the spatial coordinates. We denote such an attributed point cloud as $\{\mathbf{x}_i, \mathbf{f}_i\}_{i=1}^N$. Two point clouds $\{\mathbf{x}_i, \mathbf{f}_i\}_{i=1}^N$ and $\{\tilde{\mathbf{x}}_i, \tilde{\mathbf{f}}_i\}_{i=1}^N$ are considered *identical up to rigid motions* if there exists a bijection $\sigma : \{1, \dots, N\} \rightarrow \{1, \dots, N\}$ and a rigid motion g such that $\mathbf{f}_{\sigma(i)} = \tilde{\mathbf{f}}_i$ and $\mathbf{x}_{\sigma(i)} = g \cdot \tilde{\mathbf{x}}_i$ for all i .

Extending this concept, a *geometric graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{F})$ introduces a graph structure over the point cloud to model geometric relationships through edges. Here, \mathcal{V} is the set of nodes, \mathcal{E} the set of edges, and $\mathbf{F} = [\mathbf{f}_1, \dots, \mathbf{f}_n]$ is the matrix of node features, which may also encode geometric attributes. When edges are equipped with features, $e_{ij} \in \mathcal{E}$ denotes both the edge and its associated attributes.

Message Passing GNNs. Consider a (geometric) graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{F})$. Starting from $\mathbf{f}_i^{(0)} = \mathbf{f}_i$, message passing GNNs propagate features from iteration t to $t + 1$ as follows:

$$\begin{aligned}\mathbf{f}_i^{(t+1)} &= \text{UPD}(\mathbf{f}_i^{(t)}, \text{AGG}(\{\{\mathbf{f}_i^{(t)}, \mathbf{f}_j^{(t)}, e_{ij} \mid j \in \mathcal{N}_i\}\})), \\ \mathbf{f} &= \text{readout}(\{\{\mathbf{f}_i^{(T)} \mid i \in \mathcal{V}\}\}),\end{aligned}\tag{1}$$

where e_{ij} represents the attribute of edge (i, j) , \mathcal{N}_i denotes the neighborhood of node i , consisting of nodes in \mathcal{V} directly connected to i by an edge in \mathcal{E} , $\{\{\cdot\}\}$ denotes a multiset, and UPD, AGG, and readout are learnable functions, parameterized by multilayer perceptions.

Maximal Expressive GNNs. The expressiveness of GNNs is often analyzed through the lens of the Weisfeiler-Lehman (WL) graph isomorphism test [44, 28], which provides a theoretical foundation for distinguishing non-isomorphic graphs. A GNN is said to be *maximally expressive* if its key components, i.e., UPD, AGG, and readout, are injective [19, 40]. This notion can be viewed as pushing a given GNN architecture to its theoretical expressive limit under ideal conditions. Under this assumption, we ask whether a given GNN architecture can distinguish all non-isomorphic graph structures; that is, whether a maximally expressive GNN can produce distinct readout features for non-isomorphic graphs, given a sufficient number of layers T .

This theoretical framework has been extended to geometric graphs, particularly those derived from point clouds, where node and edge features encode spatial or geometric information. A recent study [41] investigates whether maximally expressive GNNs can distinguish point clouds—up to rigid motions—by operating on their corresponding SCHull graphs (see Appendix C for a review), specific geometric graphs constructed from these point clouds. Since our framework adopts the SCHull graph as its underlying representation, we summarize the relevant expressiveness result below:

Theorem 2.1. [41] *Let F be a maximally expressive GNN with depth $T = 1$. Then F can distinguish between the attributed SCHull graphs of any two non-isomorphic generic point clouds.*

Remark 2.2. The genericity of point clouds refers to the condition that the point coordinates are algebraically independent over the field of rational numbers—i.e., they do not satisfy any nontrivial polynomial equation with rational coefficients. This condition holds for most protein structures encountered in practice. Therefore, we assume genericity for all structures studied in this work.

Local Frames. A *local frame* is an orthogonal matrix $g \in \text{O}(3)$, consisting of three orthonormal vectors that define a local 3D coordinate system. Notably, local frames are *equivariant* under rotations and reflections: when a rotation or reflection is applied to an object’s coordinates, the associated local frame transforms accordingly. Formally, if $g(\mathbf{X})$ denotes the local frame computed from a matrix of coordinates \mathbf{X} , then for any $h \in \text{O}(3)$, we have $g(h \cdot \mathbf{X}) = h \cdot g(\mathbf{X})$. In molecular and structural modeling, spatial units such as functional groups are often associated with such local frames to capture their orientations [8, 9]. Given two spatial units with associated local frames g_i and g_j , the product $g_i^\top g_j$ represents the rotation (or reflection) that aligns one frame with the other, thereby encoding their relative orientation [9]. This formulation provides a principled way to compare geometric configurations and serves a role analogous to transition maps in differential geometry, enabling accurate geometric information to pass across local reference systems.

3 Hierarchical Graph Representations for Proteins

In this section, we present a new multiscale hierarchical graph representation for proteins. We begin by discussing the rationale for hierarchical graph construction, designed to capture protein structures at multiple levels of granularity. Next, we describe the graph construction process, detailing how it represents structural information within and across secondary structures and preserves critical spatial features at each level.

3.1 Protein Hierarchical Structures

Proteins exhibit an inherently hierarchical organization, structured across multiple scales. At the most fundamental level, they can be considered as linear sequences of amino acids, known as the primary structure. Current trends in graph-based protein modeling leverage this residue-level information by representing each amino acid as a node, with edges capturing chemical bonds and spatial proximities. This graph-based representation effectively models local interactions, reflecting the biological principles underlying protein folding.

However, the primary structure is only the first step in the hierarchical organization of proteins. As folding progresses, *contiguous sequences* of residues self-assemble into more complex geometric mo-

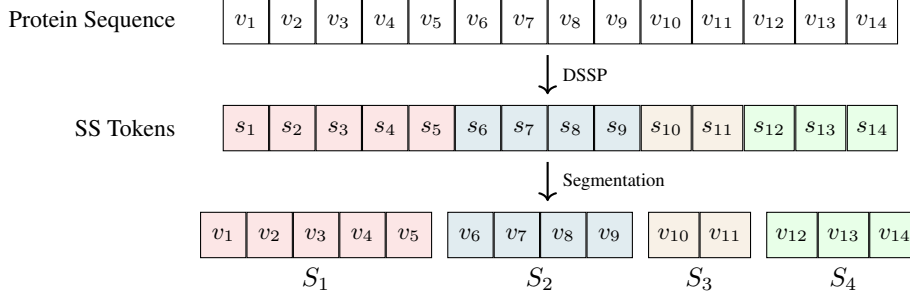


Figure 3: A visual illustration of the identification and segmentation process for protein secondary structures. Each residue v_k is assigned a secondary structure (SS) token s_k by DSSP, and consecutive residues with the same token are grouped into subsequences S_i .

tifs—such as α -helices and β -strands—forming the secondary structure. These recurring geometric patterns are stabilized through hydrogen bonding and are essential not only for maintaining the overall structure of the protein but also for determining its functions and interactions with other molecular units. This multiscale complexity underscores the necessity for a hierarchical modeling approach that can effectively capture both local and global structural features, as well as LRDs inherent in protein folding.

3.2 Identification and Segmentation of Protein Secondary Structures

The identification of secondary structures is a critical step in protein modeling, providing insights into the folding patterns and spatial organization of amino acid sequences. Various methods have been developed for this purpose, especially the DSSP (Define Secondary Structure of Proteins) algorithm [21, 35] being one of the most widely used due to its robustness and accuracy. DSSP analyzes a protein’s backbone conformation by first identifying backbone-backbone hydrogen bonds (H-bonds) based on geometric criteria and hydrogen-bond energy calculations. It then uses these H-bonds to detect structural motifs, including turns, bridges, α -helices, and β -sheets. A detailed description of the DSSP algorithm is provided in Appendix D.

Token	Secondary Structure
‘H’	α -helix
‘B’	Isolated β -bridge
‘E’	Strand (all other β -ladder residues)
‘G’	3_{10} -helix
‘I’	π -helix
‘P’	κ -helix (poly-proline II helix)
‘T’	Turn
‘S’	Bend
‘.’	None

Table 1: Secondary structure tokens and their corresponding types.

For a protein with amino acid sequence represented as $\{v_k\}_{k=1}^N$, where each v_k denotes a residue, DSSP assigns a secondary structure token s_k to each residue, indicating its structural type (e.g., α -helix, β -strand, or none). Table 1 summarizes the complete list of secondary structure tokens and their corresponding motifs. This process yields an annotated sequence $\{(v_k, s_k)\}_{k=1}^N$, where each residue is paired with its secondary structure token. We then use this annotation to segment the sequence into a set of structurally coherent subsequences, denoted by $\{S_i := \{v_k\}_{k=n_i}^{n_{i+1}-1}\}_{i=1}^I$. Each subsequence S_i consists of consecutive residues that share the same token of the secondary structure s_k . Figure 3 illustrates the process of identifying and segmenting the protein sequence based on secondary structure annotations.

The resulting set of subsequences $\{S_i\}_{i=1}^I$, grouping residues by their secondary structure types, forms the foundation for constructing the hierarchical graphs. Each subsequence—representing a distinct secondary structural motif—will serve as a node in the higher-level graph representation.

3.3 Construction of Hierarchical Geometric Graphs

Building upon segmenting a protein sequence into secondary structure-based subsequences $\{S_i\}_{i=1}^I$, we construct a hierarchical geometric graph representation that captures both fine-grained and coarse-grained relationships. Specifically, we construct a collection of *intra-structural graphs*, each modeling residue-level interactions within a secondary structure unit by treating residues as nodes, and a single *inter-structural graph* that captures the spatial organization and relative orientation among these units, treating each unit as a node. This multiscale design enables the framework to preserve detailed geometric features within motifs while capturing LRDs across the overall protein structure. An illustrative example of our hierarchical graph construction is provided in Fig. 4. To ensure geometric completeness and computational efficiency, we adopt the SCHull graph construction method [41]

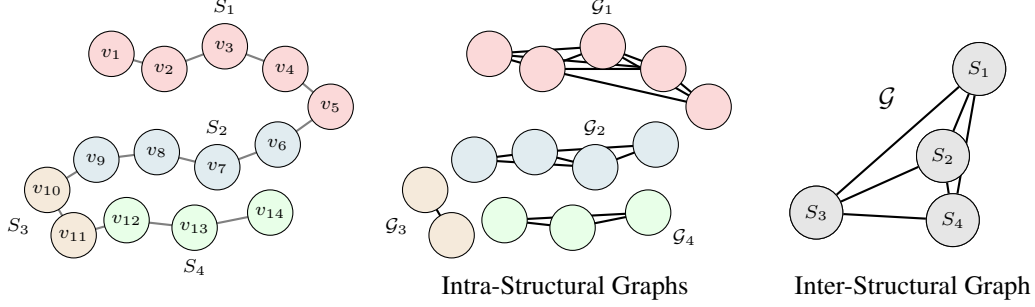


Figure 4: Hierarchical geometric graph construction. Left: A synthetic protein-like structure composed of 14 residues $\{v_k\}_{k=1}^{14}$, grouped into four secondary structure subsequences $\{S_i\}_{i=1}^4$. Middle: Intra-structural graphs G_i capture local information within each subsequence S_i using SCHull. Right: The inter-structural graph G is formed by connecting the geometric centers of each S_i , modeling higher-level structural relationships between secondary structural motifs.

(see also Appendix C), which constructs sparse yet rigid graphs based on node coordinates. We now describe how this method is applied at both levels of our hierarchical construction:

Intra-Structural Graph G_i . For each secondary structure unit S_i , we build an intra-structural graph G_i , where nodes represent residues within S_i , and edges are computed based on the 3D coordinates of their α -carbon atoms using the SCHull method. Node and edge features include geometric features generated by SCHull, combined with residue-specific attributes (e.g., amino acid type).

Inter-Structural Graph G . The inter-structural graph G is built by treating each secondary structure unit S_i as a node, with its coordinate defined by the geometric center of its residues. SCHull is again applied to determine edges between these structural units based on the spatial arrangement of their centers. In addition to geometric features on SCHull, we incorporate an additional edge feature, $g_i^\top g_j$, where $g_i = \mathcal{F}(G_i)$ denotes a local frame computed from G_i . The product $g_i^\top g_j$ captures the relative 3D orientation between two secondary structure units, following the orientation encoding proposed in [9] (see also Section 2 for details). This orientation encoding is the key to ensuring the expressiveness guarantee proved in Section 4. The construction of frames is discussed in Appendix B.

Remark 3.1. Dihedral angles are widely used to capture relative orientations between adjacent residues [20] or local structural motifs [11, 38]. In contrast, our approach leverages local frames to encode relative orientations between secondary structure units. Notably, as shown in [9], dihedral angles are inherently contained within the product $g_i^\top g_j$, which encodes richer geometric features.

Beyond geometric fidelity, our hierarchical construction also ensures strong sparsity, which is crucial for scaling GNNs to long protein sequences. The following proposition provides an upper bound on the total number of edges in the hierarchical graph:

Proposition 3.2. *Let N be the total number of residues in a protein, denoted by $\{v_k\}_{k=1}^N$, and let $\{S_i\}_{i=1}^I$ represent its segmentation into secondary structure units. For each unit S_i , let G_i denote the intra-structural graph, and let G be the inter-structural graph connecting the structural units. Let \mathcal{E}_i and \mathcal{E} denote the sets of edges in G_i and G , respectively. Then the total number of edges in the full hierarchical representation satisfies:*

$$|\mathcal{E}| + \sum_{i=1}^I |\mathcal{E}_i| < 3N.$$

In Section 5, we report the total number of edges created in our framework, along with the average runtime and memory usage, and compare these metrics against existing methods across benchmarks to highlight the efficiency gains enabled by our sparse hierarchical design.

4 A Two-Stage GNN Architecture for Multiscale Protein Modeling

Finally, we introduce a two-stage GNN framework that leverages our multiscale graph representation for efficient and expressive protein learning. Figure 2 illustrates the overall architecture.

The first-stage GNN operates independently on each intra-structural graph \mathcal{G}_i , where each graph corresponds to a single secondary structure unit. It encodes local geometric and chemical interactions among residues and generates embeddings that summarize each unit’s internal feature. These embeddings are then passed to the second-stage GNN, which treats each secondary structure unit as a node in the inter-structural graph \mathcal{G} . This graph captures spatial and functional relationships between structural motifs, enabling the second GNN to model LRDs and global features of the protein. Figure 2 provides an overview of this multiscale learning framework.

Message Passing within Secondary Structure Units. The first-stage GNN applies message passing to each intra-structural graph \mathcal{G}_i , capturing local geometric and chemical interactions and producing a compressed embedding \mathbf{s}_i for each secondary structure unit:

$$\begin{aligned} \mathbf{f}_k^{(t+1)} &= \text{UPD}_1(\mathbf{f}_k^{(t)}, \text{AGG}_1(\{\{\mathbf{f}_k^{(t)}, \mathbf{f}_l^{(t)}, \mathbf{e}_{kl} \mid l \in \mathcal{N}_k(\mathcal{G}_i)\}\})), \quad \text{for } t = 0, 1, \dots, T_1 - 1, \\ \mathbf{s}_i &= \text{readout}_1(\{\{\mathbf{f}_k^{(T_1)} \mid k \in \mathcal{V}(\mathcal{G}_i)\}\}), \end{aligned} \quad (2)$$

where $\mathbf{f}_k^{(0)}$ denotes the initial node feature of residue v_k (e.g., amino acid type) and \mathbf{e}_{kl} denotes the attribute of edge (k, l) on \mathcal{G}_i , $\mathcal{N}_k(\mathcal{G}_i)$ is the neighborhood of node k on \mathcal{G}_i , UPD_1 updates node features, AGG_1 aggregates neighbor features, and readout_1 produces the final embedding.

Message Passing across Secondary Structure Units. The second-stage GNN operates on the inter-structural graph \mathcal{G} , where each node represents a secondary structure unit. Node features are initialized using the embeddings produced by the first-stage GNN. This stage then performs message passing over \mathcal{G} and ultimately outputs the global feature vector $\mathbf{s}_{\text{global}}$, which serves as the final output of our framework:

$$\begin{aligned} \mathbf{s}_i^{(t+1)} &= \text{UPD}_2(\mathbf{s}_i^{(t)}, \text{AGG}_2(\{\{\mathbf{s}_i^{(t)}, \mathbf{s}_j^{(t)}, \mathbf{e}_{ij} \mid \mathcal{G}_j \in \mathcal{N}_{\mathcal{G}_i}\}\})), \quad \text{for } t = 0, 1, \dots, T_2 - 1, \\ \mathbf{s}_{\text{global}} &= \text{readout}_2(\{\{\mathbf{s}_i^{(T_2)} \mid i \in \mathcal{V}(\mathcal{G})\}\}), \end{aligned} \quad (3)$$

where $\mathbf{s}_i^{(0)} = \mathbf{s}_i$, \mathbf{e}_{ij} denotes the attribute of edge (i, j) on \mathcal{G} , $\mathbf{s}_{\text{global}}$ represents the final output feature of our framework. The functions UPD_2 , AGG_2 , and readout_2 denote the update, aggregation, and readout operations, respectively.

Maximal Expressiveness of Two-Stage GNN Framework. We now provide a theoretical characterization of the expressiveness of our proposed multiscale hierarchical learning framework. This analysis builds on the notion of maximal expressiveness introduced in Section 2 (see also Theorem 2.1), a standard approach for evaluating the expressiveness of GNNs [44, 28, 19]. To formalize maximal expressiveness in our setting, we begin with the following assumption:

Assumption 4.1. UPD_1 , UPD_2 , AGG_1 , AGG_2 , readout_1 , and readout_2 are injective.

This assumption is commonly adopted in theoretical analyses of GNNs to characterize the *best possible* representational power of an architecture [44, 28, 19, 40]. Crucially, it serves as a theoretical tool rather than a requirement for practical implementations, illustrating what the model could achieve under ideal conditions. Under this framework, we can formally state the following result on the maximal expressiveness of our model:

Theorem 4.2. *Let F denote the two-stage GNN architecture defined in Section 4, with depths $T_1, T_2 \geq 1$, and using the hierarchical graph construction described in Section 4. Under Assumption 4.1, F can distinguish any pair of protein structures that are not identical under rigid motions.*

Remark 4.3. In practice, we do not strictly enforce the injectivity required by Assumption 4.1, instead relying on sufficiently expressive MLPs with ReLU activations. Even when using non-injective pooling operations (e.g., mean pooling), models integrated with the SSHG framework still demonstrate consistent performance improvements, as shown in Section 5. Incorporating more expressive or injective aggregation schemes remains a promising direction for future work.

5 Numerical Experiments

We evaluate the effectiveness and efficiency of our proposed Secondary Structure-based Hierarchical Graph (SSHG) learning framework on two benchmark protein modeling tasks: enzyme reaction

classification [15] and protein-ligand binding affinity (LBA) prediction [39, 27]. Our goal is to assess the following:

Efficiency: The hierarchical design of SSHG (Figure 2) enables integrated GNNs to operate on sparse graphs, leading to a significant reduction in training time.

Efficacy: Despite operating on sparser graphs, SSHG-based models achieve superior accuracy, consistently matching or even exceeding state-of-the-art (SOTA) models. Full model configurations and dataset statistics are provided in Appendix E.2 and E.1.

Experiment Setup: All models are implemented using PyTorch Geometric [10] and trained on NVIDIA RTX 3090 GPUs. To mitigate overfitting, we follow [37] and apply Gaussian noise (std = 0.1) and anisotropic scaling in the range [0.9, 1.1] to the node coordinates in both the original graph framework and SSHG framework. Additionally, we randomly mask amino acid types and secondary structure types with probabilities of 0.1 or 0.2. We apply the SCHull graph construction method [41] to construct intra-structural and inter-structural graphs. Specific training setups, architectures, and hyperparameters for different tasks are available in Appendix E.

Baseline and Metrics: We integrate our SSHG framework with several backbone models, including GVP-GNN [18], ProNet-Backbone [37], and Mamba [13]; see Appendix E.3 for implementation details. Models enhanced with SSHG are denoted by appending “+SSHG” to the original model name (e.g., Mamba+SSHG, ProNet+SSHG). We compare these SSHG-augmented models against a range of established baselines, including GCN [22], IEConv [14], DWNN [26], GearNet [48], HoloProt [33], GVP-GNN [18], and ProNet-Backbone [37], across two benchmark tasks: enzyme reaction classification (React)[15] and protein-ligand binding affinity prediction (LBA)[39, 27]. Performance is evaluated using classification accuracy for EC reaction classification, and standard regression metrics, including root mean square error (RMSE), Pearson correlation, and Spearman correlation for LBA. To further highlight the efficiency and scalability of our framework, we also report additional metrics in the ablation study, including training time per epoch (s/epoch), memory usage, number of model parameters, and the average total number of edges in the graph representations used by the GNNs.

5.1 EC Reaction Classification

Enzymes, which catalyze biological reactions, are categorized using Enzyme Commission (EC) numbers based on the types of reactions they facilitate [29]. In this task, we evaluate the performance of our SSHG-based models—ProNet-SSHG and Mamba-SSHG—on enzyme reaction classification to demonstrate the benefits of incorporating secondary structure information and encoding geometric relationships within/across structural motifs. We follow the same dataset and experimental setup as in [37, 15]. Details of the dataset splits and training settings are provided in Appendix E.1. Notice that the baseline GVP-GNN in [18] uses a radius cutoff of 4.5 Å, achieving an accuracy of 65.5%, while we increase the cutoff to 10 Å, which improves accuracy to 68.5%

As shown in Table 2, the use of SSHG consistently improves performance across baseline models, including GCN [22], GVP-GNN [18], ProNet [37], and Mamba [13]. In particular, ProNet-SSHG significantly reduces training time compared to the original ProNet-Backbone, while matching the best-performing baseline (IEConv) with far fewer parameters (1.3M compared to 9.8M). Mamba-SSHG further improves prediction accuracy, highlighting the benefits of integrating sequence modeling into our hierarchical framework. This task confirms the advantage of SSHG in boosting both computational efficiency and classification performance. For baseline comparisons, we adopt results reported in prior works and omit training time when not provided in the original papers, except for models integrated with SSHG, which we re-evaluate using our setup to ensure fair comparison. Reported baseline results are consistent across all tasks.

Method	Test Acc	Ave.Time (s/epoch)	# params
GCN [22]	66.5	186	—
GCN+SSHG (ours)	71.2	150	—
IEConv [14]	87.2	—	9.8M
DWNN [26]	76.7	—	—
GearNet [48]	79.4	—	—
HoloProt [33]	78.9	300	1.4M
GVP-GNN [18]	68.5±0.1	334	1.0M
GVP-GNN+SSHG (ours)	73.6±0.1	236	1.0M
ProNet-Backbone [37]	86.4±0.2	210	1.3M
ProNet+SSHG (ours)	87.2±0.2	140	1.3M
Mamba [13]	85.9±0.2	236	—
Mamba+SSHG (ours)	88.4±0.3	157	1.5M

Table 2: Results of protein reaction classification. Here, “Ave.Time” denotes the average time for training one epoch.

Method	RMSE (\downarrow)	Pearson (\uparrow)	Spearman (\uparrow)	Ave.Time (s/epoch) (\downarrow)
TAPE [32]	1.890	0.338	0.286	—
IEConv [14]	1.554	0.414	0.428	—
Holoprot-Full Surface [33]	1.464	0.509	0.500	45
GCN [22]	1.925	0.322	0.287	28
GCN+SSHG (ours)	1.788	0.392	0.359	23
GVP-GNN [18]	1.529	0.441	0.432	49
GVP-GNN + SSHG (ours)	1.488	0.512	0.477	35
ProNet-Backbone [37]	1.458	0.546	0.550	32
ProNet+ SSHG (ours)	1.435 \pm 0.004	0.579 \pm 0.004	0.591 \pm 0.003	24
Mamba [13]	1.457 \pm 0.004	0.565 \pm 0.003	0.554 \pm 0.004	27
Mamba+SSHG (ours)	1.399 \pm 0.003	0.614 \pm 0.003	0.610 \pm 0.004	29

Table 3: Results of LBA prediction task. Here, ‘‘Ave.Time’’ denotes the average time for training one epoch.

5.2 Ligand Binding Affinity

We further demonstrate the effectiveness of our SSHG framework on the benchmark task of protein-ligand binding affinity (LBA) prediction. Accurate LBA prediction plays a critical role in drug discovery by guiding the selection of promising drug candidates and minimizing the need for costly and time-intensive experiments. We evaluate our models—GCN+SSHG, GVP-GNN+SSHG, ProNet-SSHG, and Mamba-SSHG—using the PDBbind dataset [39, 27], following the experimental protocol established by [18], which includes a 30% sequence identity threshold to assess model generalization to unseen proteins. More details on the dataset and experimental setup are provided in Appendix E.1. To quantify how geometric features and secondary structure information enhance the predictive capacity and generalization ability of GNNs, we evaluate model performance on the test set using standard regression metrics: RMSE, Pearson correlation, and Spearman correlation. As shown in Table 3, ProNet-SSHG outperforms all baseline models, including the best baseline², ProNet-Backbone, in terms of both predictive accuracy and computational efficiency. Mamba-SSHG further improves performance across all three metrics while maintaining competitive training speed. These results confirm the advantages of integrating SSHG into existing architectures, enabling both higher accuracy and improved scalability.

5.3 Ablation Studies

In this section, we conduct ablation studies to investigate the impact of key components in our SSHG framework. Specifically, we examine: (1) the performance and efficiency trade-offs between existing dense or sparse radial graphs versus our SSHG-based construction, (2) architectural variations in the two-stage GNN design within SSHG under comparable parameter budgets, (3) the role of the hierarchical strategy, (4) the contribution of geometric features $g_i^\top g_j$, and (5) the effect of incorporating secondary structure (SS) information. Tables 4 and 5 present results for the first two factors. Due to space constraints, analysis of the remaining components is deferred to Appendix E.5 (Table 8). All experiments are performed on the EC reaction classification task, with each model trained for 300 epochs using a batch size of 16. We report test accuracy alongside training efficiency metrics, including average time per epoch and peak memory usage.

Table 4 compares training efficiency and resource usage across different graph construction strategies. For baseline models like ProNet and GVP-GNN, increasing the radial cutoff improves accuracy but incurs substantial computational costs. Raising the cutoff from 4 to 16 increases the average number of edges from $\sim 1\text{K}$ to $\sim 15\text{K}$, leading to much higher memory usage and training time. While denser graphs enhance expressiveness, they are less practical for large-scale applications. In contrast, SSHG-based models achieve equal or better accuracy with far fewer edges and significantly lower computational overhead. This efficiency stems from the hierarchical design, which decouples local and global interactions. On both ProNet and GVP-GNN, SSHG attains up to a 2 \times speedup in training time and a 90% reduction in memory usage while still improving accuracy.

Table 5 evaluates the robustness of SSHG to architectural variations, specifically how parameters are distributed between the two GNN stages. All configurations perform well, showing the framework’s flexibility. Notably, allocating more capacity to the first-stage GNN slightly improves accuracy, suggesting that richer local (residue-level) representations are more beneficial than a larger global inter-structural stage alone in our SSHG framework.

²For these protein tasks, Mamba is implemented in our work rather than using prior implementations.

Model	+SSHG	Cutoff	Avg. Num Edges	Time (s/epoch)↓	Mem (MiB)↓	Test Acc (%)↑
ProNet	✗	4	1,034.5	138	1,290	78.1
	✗	6	4,755.2	165	7,760	82.1
	✗	8	8,013.9	185	9,580	85.6
	✗	10	11,316.8	210	14,548	86.4
	✗	16	14,881.1	247	17,768	87.0
	✓	–	1,593.3	140	1,818	87.2
GVP-GNN	✗	4	1,034.5	216	1,558	65.5
	✗	6	4,755.2	254	3,828	66.9
	✗	8	8,013.9	298	6,286	68.1
	✗	10	11,316.8	334	8,930	68.5
	✗	16	14,881.1	354	11,248	69.2
	✓	–	1,593.3	236	1,416	73.6

Table 4: **Efficiency comparison.** Training efficiency and accuracy of different GNNs with and without SSHG across varying cutoff radii. SSHG achieves higher accuracy while substantially reducing runtime and memory usage.

	MPGNN1 # params	MPGNN2 # params	Ave. Time (s/epoch)	Mem(MiB)	Test Acc
ProNet+SSHG	0.69M	0.69M	140	1818	87.2
	1.03M	0.34M	136	2656	87.4
	0.34M	1.03M	142	1720	87.1
GVP-GNN+SSHG	0.53M	0.53M	236	1416	73.6
	0.79M	0.27M	232	1451	75.3
	0.27M	0.79M	228	1372	71.6

Table 5: **Architecture Comparison:** Two-stage GNNs with varying size ratios between the first- (MPGNN1) and second-stage (MPGNN2) graph networks.

6 Concluding Remarks

In summary, we propose a multiscale and scalable GNN-based framework for protein representation and learning by leveraging a hierarchical graph construction that aligns naturally with biological structures. By combining domain knowledge of secondary motifs with a multiscale graph design, our approach captures both fine-grained residue-level interactions and coarse-grained structural relationships through a collection of intra-structural graphs, each corresponding to a secondary structure motif, and a single inter-structural graph that encodes their spatial arrangement and relative orientation. Theoretically, we establish that our framework preserves maximal expressiveness, ensuring no loss of critical geometric information. Empirically, we demonstrate consistent improvements in both predictive accuracy and computational efficiency across standard benchmarks. These results highlight the potential of our method as a general and flexible foundation for protein-based learning tasks, opening up new avenues for integrating biological priors into geometric deep learning.

In future work, we plan to extend our investigation beyond the current experiments on enzyme classification and ligand-binding affinity prediction. We aim to evaluate the framework on additional tasks such as fold classification and protein–protein interaction prediction. Moreover, we will explore architectural enhancements through injective aggregation schemes, more expressive pooling mechanisms, alternative motif definitions, and integration with pretrained protein language models to further improve the framework’s generality and performance.

Societal Impacts: Our paper presents a new efficient and accurate machine learning model for learning biomolecules, which can impact structural biology and life sciences. We do not see additional negative societal impact compared to existing approaches due to our work.

Acknowledgement

This material is based on research sponsored by NSF grants DMS-2152762, DMS-2208361, DMS-2219956, DMS-2208356, and DMS-2436344, and DOE grants DE-SC0023490, DE-SC0025589, and DE-SC0025801. This work is also supported by NIH grant R01HL16351.

References

- [1] Luciano A Abriata. The Nobel Prize in chemistry: past, present, and future of AI in biology. *Communications Biology*, 7(1):1409, 2024.
- [2] Justin Baker, Shih-Hsin Wang, Tommaso de Fernex, and Bao Wang. An explicit frame construction for normalizing 3D point clouds. In *Forty-first International Conference on Machine Learning*, 2024.
- [3] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, dec 1996.
- [4] Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.
- [5] Byron Caughey, Heidi G Standke, Efrosini Artikis, Forrest Hoyt, and Allison Kraus. Pathogenic prion structures at high resolution. *PLoS Pathogens*, 18(6):e1010594, 2022.
- [6] Jose M Dana, Aleksandras Gutmanas, Nidhi Tyagi, Guoying Qi, Claire O’Donovan, Maria Martin, and Sameer Velankar. Sifts: updated structure integration with function, taxonomy and sequences resource allows 40-fold increase in coverage of structure-based annotations for proteins. *Nucleic acids research*, 47(D1):D482–D489, 2019.
- [7] Tri Dao, Shizhe Ma, Wenxuan Sun, Albert Gu, Sam Smith, Aapo Kyrola, Christopher D Manning, and Christopher Re. An empirical study of state space models for large language modeling. *arXiv preprint arXiv:2406.07887*, 2024.
- [8] Weitao Du, He Zhang, Yuanqi Du, Qi Meng, Wei Chen, Nanning Zheng, Bin Shao, and Tie-Yan Liu. Se (3) equivariant graph neural networks with complete local frames. In *International Conference on Machine Learning*, pages 5583–5608. PMLR, 2022.
- [9] Yuanqi Du, Limei Wang, Dieqiao Feng, Guifeng Wang, Shuiwang Ji, Carla P Gomes, Zhi-Ming Ma, et al. A new perspective on building efficient and expressive 3d equivariant graph neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.
- [10] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.
- [11] Johannes Gasteiger, Florian Becker, and Stephan Günnemann. Gemnet: Universal directional graph neural networks for molecules. *Advances in Neural Information Processing Systems*, 34:6790–6802, 2021.
- [12] Vladimir Gligoričević, P Douglas Renfrew, Tomasz Kosciółek, Julia Koehler Leman, Daniel Berenberg, Tommi Vatanen, Chris Chandler, Bryn C Taylor, Ian M Fisk, Hera Vlamakis, et al. Structure-based protein function prediction using graph convolutional networks. *Nature communications*, 12(1):3168, 2021.
- [13] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.
- [14] Pedro Hermosilla, Marco Schäfer, Matěj Lang, Gloria Fackelmann, Pere Pau Vázquez, Barbora Kozlíková, Michael Krone, Tobias Ritschel, and Timo Ropinski. Intrinsic-extrinsic convolution and pooling for learning on 3d protein structures. *arXiv preprint arXiv:2007.06252*, 2020.
- [15] Jie Hou, Badri Adhikari, and Jianlin Cheng. Deepsf: deep convolutional neural network for mapping protein sequences to folds. *Bioinformatics*, 34(8):1295–1303, 2018.

- [16] Bozhen Hu, Cheng Tan, Yongjie Xu, Zhangyang Gao, Jun Xia, Lirong Wu, and Stan Z Li. Protgo: Function-guided protein modeling for unified representation learning. *Advances in Neural Information Processing Systems*, 37:88581–88604, 2024.
- [17] Bowen Jing, Stephan Eismann, Pratham N Soni, and Ron O Dror. Equivariant graph neural networks for 3d macromolecular structure. *arXiv preprint arXiv:2106.03843*, 2021.
- [18] Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael JL Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. In *International Conference on Learning Representations*, 2021.
- [19] Chaitanya K Joshi, Cristian Bodnar, Simon V Mathis, Taco Cohen, and Pietro Lio. On the expressive power of geometric graph neural networks. In *International conference on machine learning*, pages 15330–15355. PMLR, 2023.
- [20] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon Kohl, Andrew Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.
- [21] Wolfgang Kabsch and Christian Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, 1983.
- [22] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.
- [23] Allison Kraus, Forrest Hoyt, Cindi L Schwartz, Bryan Hansen, Efrosini Artikis, Andrew G Hughson, Gregory J Raymond, Brent Race, Gerald S Baron, and Byron Caughey. High-resolution structure and strain comparison of infectious mammalian prions. *Molecular cell*, 81(21):4540–4551, 2021.
- [24] Allison Kraus, Forrest Hoyt, Cindi L Schwartz, Bryan Hansen, Andrew G Hughson, Efrosini Artikis, Brent Race, and Byron Caughey. Structure of an infectious mammalian prion. *BioRxiv*, pages 2021–02, 2021.
- [25] AI21 Labs. Jamba: Open source hybrid ssm + transformer llm. <https://www.ai21.com/blog/introducing-jamba>, 2024. Accessed: 2025-04-17.
- [26] Jiahua Li. Directed weight neural networks for protein structure representation learning. *arXiv preprint arXiv:2201.13299*, 2022.
- [27] Zhihai Liu, Yan Li, Li Han, Jie Li, Jie Liu, Zhixiong Zhao, Wei Nie, Yuchen Liu, and Renxiao Wang. Pdb-wide collection of binding data: current status of the pdbname database. *Bioinformatics*, 31(3):405–412, 2015.
- [28] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.
- [29] Marina V Omelchenko, Michael Y Galperin, Yuri I Wolf, and Eugene V Koonin. Non-homologous isofunctional enzymes: a systematic analysis of alternative solutions in enzyme evolution. *Biology direct*, 5:1–20, 2010.
- [30] Perrakis Group. Dssp: Define secondary structure of proteins. <https://pdb-redo.eu/dssp/about>. [Online; accessed 20-January-2025].

- [31] Ruijie Quan, Wenguan Wang, Fan Ma, Hehe Fan, and Yi Yang. Clustering for protein representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 319–329, 2024.
- [32] Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Xi Chen, John Canny, Pieter Abbeel, and Yun S. Song. Evaluating protein transfer learning with tape, 2019.
- [33] Vignesh Ram Somnath, Charlotte Bunne, and Andreas Krause. Multi-scale representation learning on proteins. *Advances in Neural Information Processing Systems*, 34:25244–25255, 2021.
- [34] Wenxuan Sun, Tri Dao, Hongyu Zhuang, Zihang Dai, Albert Gu, and Christopher D Manning. Llama: Efficient llms with mamba-based distillation. *arXiv preprint arXiv:2502.14458*, 2024.
- [35] Wouter G Touw, Coos Baakman, Jon Black, Tim AH Te Beek, Elmar Krieger, Robbie P Joosten, and Gert Vriend. A series of pdb-related databanks for everyday needs. *Nucleic acids research*, 43(D1):D364–D368, 2015.
- [36] Chao Wang, Hehe Fan, Ruijie Quan, and Yi Yang. Protchatgpt: Towards understanding proteins with large language models. *arXiv preprint arXiv:2402.09649*, 2024.
- [37] Limei Wang, Haoran Liu, Yi Liu, Jerry Kurtin, and Shuiwang Ji. Learning hierarchical protein representations via complete 3d graph networks. *arXiv preprint arXiv:2207.12600*, 2022.
- [38] Limei Wang, Yi Liu, Yuchao Lin, Haoran Liu, and Shuiwang Ji. Comenet: Towards complete and efficient message passing for 3d molecular graphs. *Advances in Neural Information Processing Systems*, 35:650–664, 2022.
- [39] Renxiao Wang, Xueliang Fang, Yipin Lu, and Shaomeng Wang. The pdbind database: Collection of binding affinities for protein- ligand complexes with known three-dimensional structures. *Journal of medicinal chemistry*, 47(12):2977–2980, 2004.
- [40] Shih-Hsin Wang, Yung-Chang Hsu, Justin Baker, Andrea L. Bertozzi, Jack Xin, and Bao Wang. Rethinking the benefits of steerable features in 3d equivariant graph neural networks. In *The Twelfth International Conference on Learning Representations*, 2024.
- [41] Shih-Hsin Wang, Yuhao Huang, Justin M Baker, Yuan-En Sun, Qi Tang, and Bao Wang. A theoretically-principled sparse, connected, and rigid graph representation of molecules. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [42] Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.
- [43] Fang Wu, Dragomir Radev, and Stan Z Li. Molformer: Motif-based transformer on 3d heterogeneous molecular graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5312–5320, 2023.
- [44] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.
- [45] Minghao Xu, Xinyu Yuan, Santiago Miret, and Jian Tang. Protst: Multi-modality learning of protein sequences and biomedical texts. In *International Conference on Machine Learning*, pages 38749–38767. PMLR, 2023.
- [46] Zhaoning Yu and Hongyang Gao. Molecular representation learning via heterogeneous motif graph neural networks. In *International conference on machine learning*, pages 25581–25594. PMLR, 2022.
- [47] Zuobai Zhang, Pascal Notin, Yining Huang, Aurélie Lozano, Vijil Chenthamarakshan, Debora Marks, Payel Das, and Jian Tang. Multi-scale representation learning for protein fitness prediction. *arXiv preprint arXiv:2412.01108*, 2024.
- [48] Zuobai Zhang, Minghao Xu, Arian Jamasb, Vijil Chenthamarakshan, Aurelie Lozano, Payel Das, and Jian Tang. Protein representation learning by geometric structure pretraining. *arXiv preprint arXiv:2203.06125*, 2022.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [NA].
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes]" is generally preferable to "[No]", it is perfectly acceptable to answer "[No]" provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No]" or "[NA]" is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading “NeurIPS Paper Checklist”,**
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers.**

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

Answer: [Yes]

Justification: The abstract summarizes our theoretical and algorithmic contributions in this paper.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: In the conclusion section, we have listed a few potential future works.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: All assumptions are commonly used by the community, and we have discussed where the assumptions come from.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: See the experimental setup.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have submitted the code and data as a supplementary file.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: See the experimental setup.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[Yes\]](#)

Justification: We have provided the standard deviation.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: See details in the numerical experiments section.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines?>

Answer: [Yes]

Justification: We fully comply with this guideline.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: See the conclusion section.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [No]

Justification: All data used are standard.

Guidelines:

- The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [\[Yes\]](#)

Justification: We have acknowledged the codes we have used.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [\[Yes\]](#)

Justification: The codes have been well-documented.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [\[NA\]](#)

Justification: Our work does not include this.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our research does not include this.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [No]

Justification: We only use LLM for formatting purposes.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

Appendices

A Missing proofs

Proposition 3.2. *Let N be the total number of residues in a protein, denoted by $\{v_k\}_{k=1}^N$, and let $\{S_i\}_{i=1}^I$ represent its segmentation into secondary structure units. For each unit S_i , let \mathcal{G}_i denote the intra-structural graph, and let \mathcal{G} be the inter-structural graph connecting the structural units. Let \mathcal{E}_i and \mathcal{E} denote the sets of edges in \mathcal{G}_i and \mathcal{G} , respectively. Then the total number of edges in the full hierarchical representation satisfies:*

$$|\mathcal{E}| + \sum_{i=1}^I |\mathcal{E}_i| < 3N.$$

Proof of Proposition 3.2. Recall that for a point cloud with $m > 2$ points, the SCHull algorithm constructs a geometric graph with at most $3m - 6$ edges [41]. For the remaining cases, when $m = 1$, no edges can be formed, and when $m = 2$, there is exactly one edge connecting the two points.

Now, consider the following partition of $\{1, 2, \dots, I\}$:

$$\begin{aligned} J_1 &= \{i \mid |S_i| = 1\}, \\ J_2 &= \{i \mid |S_i| = 2\}, \\ J_{\geq 3} &= \{i \mid |S_i| \geq 3\}. \end{aligned} \tag{4}$$

According to the construction of S_i , we have

$$|J_1| + |J_2| + |J_{\geq 3}| = I \text{ and } |J_1| + 2|J_2| + \sum_{i \in J_{\geq 3}} |S_i| = N. \tag{5}$$

Then we have for each intra-structural graph \mathcal{G}_i , constructed over the residues in secondary structure unit S_i , the number of edges satisfies:

$$|\mathcal{E}_i| \leq \begin{cases} 0 & \text{if } i \in J_1 \\ 1 & \text{if } i \in J_2 \\ 3|S_i| - 6 & \text{if } i \in J_{\geq 3} \end{cases} \tag{6}$$

Summing over all intra-structural graphs gives:

$$\begin{aligned} \sum_{i=1}^I |\mathcal{E}_i| &= \sum_{i \in J_1} |\mathcal{E}_i| + \sum_{i \in J_2} |\mathcal{E}_i| + \sum_{i \in J_{\geq 3}} |\mathcal{E}_i| \\ &= 0 \cdot |J_1| + 1 \cdot |J_2| + \sum_{i \in J_{\geq 3}} (3|S_i| - 6) \\ &= |J_2| + 3(N - |J_1| - 2|J_2|) - 6|J_{\geq 3}| \\ &\leq 3(N - |J_1| - |J_2| - |J_{\geq 3}|) \\ &\leq 3(N - I). \end{aligned} \tag{7}$$

For the inter-structural graph \mathcal{G} , which connects the I secondary structure units, SCHull yields:

$$|\mathcal{E}| \leq 3I - 6.$$

Combining both bounds:

$$|\mathcal{E}| + \sum_{i=1}^I |\mathcal{E}_i| \leq (3I - 6) + 3(N - I) = 3N - 6 < 3N.$$

Therefore, the total number of edges in the hierarchical representation is strictly less than $3N$, completing the proof. \square

Before proving Theorem 4.2, we first recall Theorem 2.1, which will be applied in the argument:

Theorem 2.1. [41] *Let F be a maximally expressive GNN with depth $T = 1$. Then F can distinguish between the attributed SCHull graphs of any two non-isomorphic generic point clouds.*

Theorem 4.2. *Let F denote the two-stage GNN architecture defined in Section 4, with depths $T_1, T_2 \geq 1$, and using the hierarchical graph construction described in Section 4. Under Assumption 4.1, F can distinguish any pair of protein structures that are not identical under rigid motions.*

Proof of Theorem 4.2. Suppose the model F assigns identical outputs to the hierarchical graphs of two protein structures represented by the point clouds (\mathbf{X}, \mathbf{F}) and $(\mathbf{X}', \mathbf{F}')$. Let $\mathcal{G}, \mathcal{G}'$ denote their respective hierarchical graphs, and let $\mathcal{G}_i, \mathcal{G}'_j$ denote the intra-structural graphs corresponding to their secondary structure units. Denote the first- and second-stage GNNs in F by F_1 and F_2 , respectively.

By construction, the inter-structural graph is defined over the set of tuples $\{(F_1(\mathcal{G}_i), \mathbf{z}_i)\}_{i=1}^I$ and $\{(F_1(\mathcal{G}'_j), \mathbf{z}'_j)\}_{j=1}^J$, where \mathbf{z}_i and \mathbf{z}'_j are the geometric centers of the secondary structure units \mathcal{G}_i and \mathcal{G}'_j , respectively. According to Theorem 2.1, F_2 assigns identical outputs to \mathcal{G} and \mathcal{G}' if and only if $\{(F_1(\mathcal{G}_i), \mathbf{z}_i)\}_{i=1}^I$ and $\{(F_1(\mathcal{G}'_j), \mathbf{z}'_j)\}_{j=1}^J$ are identical up to a rigid motion. That is, there exists a bijection $b : \{1, \dots, I\} \rightarrow \{1, \dots, J\}$ and a rigid transformation g such that for all i ,

$$F_1(\mathcal{G}_i) = F_1(\mathcal{G}'_{b(i)}), \quad \text{and} \quad \mathbf{z}_i = g \cdot \mathbf{z}'_{b(i)}.$$

Reindexing the units according to the bijection, we can assume: $F_1(\mathcal{G}_i) = F_1(\mathcal{G}'_i)$ for all i . Additionally, our construction includes the relative frame feature $g_i^\top g_j$ as an edge attribute. Since these are preserved between \mathcal{G} and \mathcal{G}' , we have

$$g_i^\top g_j = g_i'^\top g_j' \quad \text{for all } i, j,$$

where g_i denotes the local frame of \mathcal{G}_i , constructed using the method described in Appendix B. Specifically, each frame g_i comprises an orthogonal matrix representing the orientation and a vector specifying the geometric center. By Lemma A.1, the equality $(F_1(\mathcal{G}_i), F_1(\mathcal{G}_j), g_i^\top g_j) = (F_1(\mathcal{G}'_i), F_1(\mathcal{G}'_j), g_i'^\top g_j')$ implies that the underlying point clouds of $\mathcal{G}_i \cup \mathcal{G}_j$ and $\mathcal{G}'_i \cup \mathcal{G}'_j$ are identical up to an isometry. Using an inductive argument, we conclude that the union of all point clouds across the units \mathcal{G}_i and \mathcal{G}'_i must also be identical up to an isometry. Thus, the full point clouds (\mathbf{X}, \mathbf{F}) and $(\mathbf{X}', \mathbf{F}')$ must be identical up to a global isometry. \square

Lemma A.1. *The 3-tuple $(F_1(\mathcal{G}_i), F_1(\mathcal{G}_j), g_i^\top g_j)$ uniquely determines the union of the underlying point clouds of $\mathcal{G}_i, \mathcal{G}_j$ up to an isometry.*

Proof. Let $\mathcal{P}_i := \{(\mathbf{x}_{i,k}, \mathbf{f}_{i,k})\}$ and $\mathcal{P}_j := \{(\mathbf{x}_{j,k}, \mathbf{f}_{j,k})\}$ denote the underlying point clouds of \mathcal{G}_i and \mathcal{G}_j , respectively. Let \mathcal{F} be the function used to generate the equivariant local frame for each secondary structure unit, as described in Appendix B. That is, $g_i = \mathcal{F}(\mathcal{G}_i) = \mathcal{F}(\mathcal{P}_i)$ for any i . Our goal is to show that the union $\mathcal{P}_i \cup \mathcal{P}_j$ is uniquely determined, up to an isometry, by the 5-tuple $(F_1(\mathcal{G}_i), \mathbf{z}_i, F_1(\mathcal{G}_j), \mathbf{z}_j, g_i^\top g_j)$.

We first observe that the function c , which maps the point cloud \mathcal{P}_i to $\mathcal{F}(\mathcal{G}_i)^\top \cdot \mathcal{P}_i := \{(\mathcal{F}(\mathcal{G}_i)^\top \cdot \mathbf{x}_{i,k}, \mathbf{f}_{i,k})\}$, is invariant under any rigid motion applied to $\{\mathbf{x}_{i,k}\}$. To see this, consider a rigid motion $g \in \text{E}(3)$, where $\text{E}(3)$ denotes the Euclidean group of isometries. By the equivariance of \mathcal{F} , we have:

$$(\mathcal{F}(g \cdot \mathcal{G}_i))^\top \cdot g \cdot \mathbf{x}_{i,k} = (\mathcal{F}(\mathcal{G}_i))^\top g^\top g \cdot \mathbf{x}_{i,k} = \mathcal{F}(\mathcal{G}_i)^\top \cdot \mathbf{x}_{i,k},$$

which confirms that c is invariant.

By Theorem 2.1, there exists a function h such that $c(\mathcal{P}_i) = h(F_1(\mathcal{G}_i))$ for any point cloud. Now consider the following construction that combines the information in the 3-tuple.

$$\begin{aligned} h(F_1(\mathcal{G}_i)) \cup (g_i^\top g_j \cdot h(F_1(\mathcal{G}_j))) &= c(\mathcal{P}_i) \cup (g_i^\top g_j \cdot c(\mathcal{P}_j)) \\ &= (g_i^\top \mathcal{P}_i) \cup (g_i^\top g_j \cdot g_j^\top \mathcal{P}_j) \\ &= g_i^\top \cdot (\mathcal{P}_i \cup \mathcal{P}_j). \end{aligned} \tag{8}$$

This final expression shows that the union $\mathcal{P}_i \cup \mathcal{P}_j$ is uniquely determined up to a global isometry by the 3-tuple data, completing the proof. \square

B Construction of Local Frames

In [8, 9], the coordinates of nodes \mathbf{x}_i and \mathbf{x}_j on an edge (i, j) are used to define equivariant local frames, with the following transformation:

$$(\mathbf{x}_i, \mathbf{x}_j) \mapsto \left[\frac{\mathbf{x}_i}{\|\mathbf{x}_i\|}, \frac{\mathbf{x}_j}{\|\mathbf{x}_j\|}, \frac{\mathbf{x}_i \times \mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|} \right].$$

This transformation creates a local frame for the edge, converting equivariant features within a node’s neighborhood into invariant features (independent of orientation), or vice versa, through multiplication by the frame or its inverse. It can also be used to compute the crucial transition information $g_i^\top g_j$ for neighborhoods \mathcal{N}_i and \mathcal{N}_j .

However, this method is limited to pairs of nodes and cannot be directly extended to structural units, which typically consist of more than two unordered nodes. To address this, we adopt the approach in [2], which constructs equivariant frames for point clouds (or their corresponding geometric graphs) of arbitrary size. Specifically, the algorithm in [2] constructs a frame $\mathcal{F} : \mathcal{X} \rightarrow \text{E}(3)$ over the set of point clouds \mathcal{X} . This mapping produces not only an orthogonal matrix representing the orientation but also the geometric center of the point cloud, jointly forming a complete frame in Euclidean space. Importantly, the frame construction satisfies equivariance with respect to isometries: for any rigid motion $g \in \text{E}(3)$ and any point cloud $\mathcal{P} \in \mathcal{X}$, we have

$$\mathcal{F}(g \cdot \mathcal{P}) = g \cdot \mathcal{F}(\mathcal{P}),$$

where \cdot denotes the group product in $\text{E}(3)$.

Remark B.1. While this construction applies to all point clouds, it may not be fully equivariant but rather *relaxed-equivariant* for symmetric inputs. This does not impact the distinguishability of the framework, as shown in Theorem 4.2, but it could reduce the framework’s strict invariance. Fortunately, symmetric inputs are rare in practice—particularly for individual protein structures—and this issue can be mitigated by introducing small perturbations to break the symmetry.

C SCHull Graphs

In this section, we provide a brief review of the SCHull algorithm—proposed in [41]—for constructing a sparse, connected, and rigid for a given point cloud (\mathbf{X}, \mathbf{F}) be a point cloud, where $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$, \mathbf{F} denote the point coordinates and features, respectively. Let \mathcal{V} denote the point index set. In particular, the SCHull graph for the point cloud $(\mathcal{V}, \mathbf{X})$ is constructing in the following three steps:

- **Step 1: Project points onto the unit sphere.** Let

$$\bar{\mathbf{x}} := \frac{1}{N} \sum \mathbf{x}_i$$

be the center of point cloud. Consider the projection

$$p_{\bar{\mathbf{x}}} : \mathbb{R}^3 \rightarrow \mathbb{S}^2 : \mathbf{x} \mapsto \frac{\mathbf{x} - \bar{\mathbf{x}}}{\|\mathbf{x} - \bar{\mathbf{x}}\|},$$

where $\mathbb{S}^2 := \{\mathbf{x} \in \mathbb{R}^3 \mid \|\mathbf{x}\| = 1\}$ is the unit sphere. That is, $p_{\bar{\mathbf{x}}}$ projects points onto the unit sphere centered at $\bar{\mathbf{x}}$. Applying this projection to all points, we obtain a new point cloud $(\mathcal{V}, p_{\bar{\mathbf{x}}}(\mathbf{X}))$ on \mathbb{S}^2 , where

$$p_{\bar{\mathbf{x}}}(\mathbf{X}) = [p_{\bar{\mathbf{x}}}(\mathbf{x}_1), p_{\bar{\mathbf{x}}}(\mathbf{x}_2), \dots, p_{\bar{\mathbf{x}}}(\mathbf{x}_N)].$$

- **Step 2: Construct the convex hull of the projected point cloud.** Next, SCHull constructs a convex hull—using the QuickHull algorithm [3]—for the projected point cloud $(\mathcal{V}, p_{\bar{\mathbf{x}}}(\mathbf{X}))$. Notice that this step is very efficient with a computational complexity $\mathcal{O}(N \log N)$.
- **Step 3: Construct the SCHull graph.** The SCHull graph for the given point cloud (\mathbf{X}, \mathbf{F}) is then defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{F}')$. Specifically, nodes i, j are connected by an edge in \mathcal{E} if and only if their projected points on the unit sphere $p_{\bar{\mathbf{x}}}(\mathbf{x}_i), p_{\bar{\mathbf{x}}}(\mathbf{x}_j)$ are connected by an

edge on the convex hull. In addition, the graph incorporates geometric attributes as follows. Each node feature in \mathbf{F}' includes the original feature from \mathbf{F} , augmented with a scalar node attribute defined below. Similarly, each edge in \mathcal{E} is associated with the following attributes:

$$\begin{aligned} &\text{the edge attributes of } (i, j) : (\|\mathbf{x}_i - \mathbf{x}_j\|, \tau_{ij}) \text{ for any } (i, j) \in \mathcal{E}, \text{ and} \\ &\text{the node attributes} : \|\mathbf{x}_i - \bar{\mathbf{x}}\| \text{ for any } i \in \mathcal{V}. \end{aligned} \quad (9)$$

SCHull graph has two remarkable properties with provable guarantees: (1) The graph is sparse and connected with edges that satisfy $|\mathcal{E}| \leq 3N - 6$ when $N = |\mathcal{V}| \geq 3$ (cf. [41, Proposition 3.1]). (2) SCHull graphs of any two non-isomorphic generic point clouds can be distinguished by a maximally expressive GNN with depth 1 (see Theorem 2.1, i.e., [41, Theorem 3.6]).

D DSSP Algorithm

DSSP is both a database of secondary structure assignments for all protein entries in the Protein Data Bank (PDB) [4] as well as a program that calculates DSSP entries from PDB entries [21, 35]. The algorithm first detects the presence of backbone-backbone hydrogen bonds (H-bonds). An H-bond between amino acids is considered to be present if the electrostatic interaction energy, E , between the carboxyl group of one and the amino group of another is calculated to be less than -0.5 kcal/mol. Specifically, $E = 0.084 \left[\frac{1}{r(ON)} + \frac{1}{r(CH)} - \frac{1}{r(OH)} - \frac{1}{r(CN)} \right] \cdot 332$ kcal/mol where $r(AB)$ is the interatomic distance between A and B. Following [21], $\text{Hbond}(i, j)$ denotes that an H-Bond is present between the carboxyl group of residue i and the amino group of residue j .

Once H-bond presence is decided, the algorithm determines the presence of elementary H-bond patterns: n -turns (where $n=3, 4$, or 5) and bridges (which can be parallel or antiparallel). An n -turn is considered to exist at residue i if $\text{Hbond}(i, i+n)$ is present. A bridge may exist between two non-overlapping stretches of three residues each. A parallel bridge is said to be present if $\text{Hbond}(i-1, j)$ and $\text{Hbond}(j, i+1)$ or if $\text{Hbond}(j-1, i)$ and $\text{Hbond}(i, j+1)$. An antiparallel bridge is said to be present if $\text{Hbond}(i, j)$ and $\text{Hbond}(j, i)$ or if $\text{Hbond}(i-1, j+1)$ and $\text{Hbond}(j-1, i+1)$.

These patterns are then used to identify cooperative H-bond patterns: helices, β -ladders, and β -sheets. Helices consist of two consecutive n -turns for fixed n , e.g., a 4-helix is present from residue i to $i+3$ if there is a 4-turn at residue $i-1$ and another at residue i . Note that a 3-helix is commonly called a 3_{10} -helix, a 4-helix an α -helix, and a 5-helix a π -helix. β -ladders consist of one or more consecutive bridges of identical type and β -sheets consist of one or more ladders connected by shared residues. A group of five residues with high curvature is known as bend. The curvature at residue i is calculated as the angle between the backbone direction of the first three and last three residues of this group of five. Specifically, if C_j^α is the position vector of the α -carbon of residue j , a bend is considered to exist if the angle between $C_i^\alpha - C_{i-2}^\alpha$ and $C_{i+2}^\alpha - C_i^\alpha$ is greater than 70° .

Although it is possible for an amino acid to belong to more than one of these structures, each residue is assigned a single letter from the list in Table 1. The original algorithm assigned letters in the following order from left to right: H, B, E, G, I, T, S; once a residue is assigned a letter, it is not changed. More recent versions assign π -helices before α -helices [35] and also detect another type of helix known as a κ -helix or a poly-proline II (PPII) helix [30].

E Additional Experiments Details

E.1 Datasets and Experiment Overview

E.1.1 Datasets

Reaction dataset. For the reaction classification task, 3D structures of 37,428 proteins corresponding to 384 enzyme commission (EC) numbers are obtained from the Protein Data Bank, with EC annotations for each protein retrieved from the SIFTS database [6]. The dataset is divided into 29,215 proteins for training, 2,562 for validation, and 5,651 for testing. Each EC number is represented across all three splits, and protein chains sharing more than 50% sequence similarity are grouped.

LBA dataset. Following [18], we perform ligand binding affinity predictions on a subset of the commonly-used PDBbind refined set [39, 27]. The curated dataset of 3,507 complexes is split into

train/val/test splits based on a 30% sequence identity threshold to verify the model generalization ability for unseen proteins. For a protein-ligand complex, we predict the negative log-transformed binding affinity $pK = -\log_{10}(K)$ in molar units.

E.1.2 Experiment Overview

We primarily follow the GNN architectures, training setups, and hyperparameter search spaces used in the baseline models GVP-GNN [18] and ProNet [37]. Our SSHG model adopts nearly identical feature embedding functions, message passing blocks, and readout functions for the hierarchical geometric graphs—namely, the Intra-Structural Graph and Inter-Structural Graph. Furthermore, we integrate Mamba [13] to capture the sequential information of the tokens in the Inter-Structural Graph. See E.2 and E.3 for details.

E.2 Model Configuration

We integrate our SSHG framework with two models tailored for protein tasks, one is GVPNet [18] and the other is ProNet [37]. Moreover, to capture the sequential information of the secondary structure tokens, we integrate Mamba [13] into our SSHG model. See E.3.1 for further details. The illustration of the architectures of our SSHG model is shown in Figure 5. Below are some details:

- **Message Passing Blocks:** We use the same message-passing GNN (MPGNN) architectures as those in the baseline models [18, 37].
- **Edge Feature Function:** We use the same edge feature construction methods as those in the baseline models [18, 37].
- **Scatter:** The tensor output from the message passing blocks contains embedding vectors for all nodes across all graphs in the batch. We use the PyTorch scatter function to aggregate these node embeddings into a tensor with one embedding per graph, corresponding to the number of graphs in the batch. Similarly, we apply scatter to aggregate intra-structural graph node embeddings into a tensor with one embedding per unit in the inter-structural graph.

E.3 Implementation Details

E.3.1 Integrate Mamba into SSHG

To capture the sequential dependencies among secondary structure tokens in our SSHG model, we consider using Mamba. Mamba [13] is a special type of state space model defined by the following ODE system:

$$\begin{aligned}\frac{d\mathbf{h}(t)}{dt} &= \mathbf{\Lambda}(\mathbf{x}(t))\mathbf{h}(t) + \mathbf{B}(\mathbf{x}(t))\mathbf{x}(t) \\ \mathbf{y}(t) &= \mathbf{W}_o\mathbf{h}(t)\end{aligned}\tag{10}$$

where:

- t denotes the time step (discrete or continuous).
- \mathbf{x}_t denotes the feature vector at time step t in the input feature sequence.
- $\mathbf{h}(t)$ denotes the state vector, where the IC $\mathbf{h}(0)$ is often a learnable vector.
- $\mathbf{\Lambda}(\mathbf{x}(t)) := \text{diag}[\sigma(\mathbf{W}_\lambda\mathbf{x}(t)) - \mathbf{1}]/dt$ is the input-conditioned decay/filter coefficient, where \mathbf{W}_λ is time-invariant learnable parameter and σ denotes the sigmoid function.
- $\mathbf{B}(\mathbf{x}(t)) := \mathbf{W}_B\mathbf{x}(t)/dt$, where \mathbf{W}_B is time-invariant learnable parameter.
- \mathbf{W}_o is time-invariant learnable parameter.

It has been increasingly adopted in large language models (LLMs) [34, 25, 7] due to its efficiency and competitive performance compared to traditional Transformer architectures.

SSHG+Mamba: Let $\mathbf{x}_i := s_i^{(0)} = \text{readout}_1(\{\mathbf{f}_k^{(T_1)} \mid k \in \mathcal{V}(\mathcal{G}_i)\})$ be the initial node

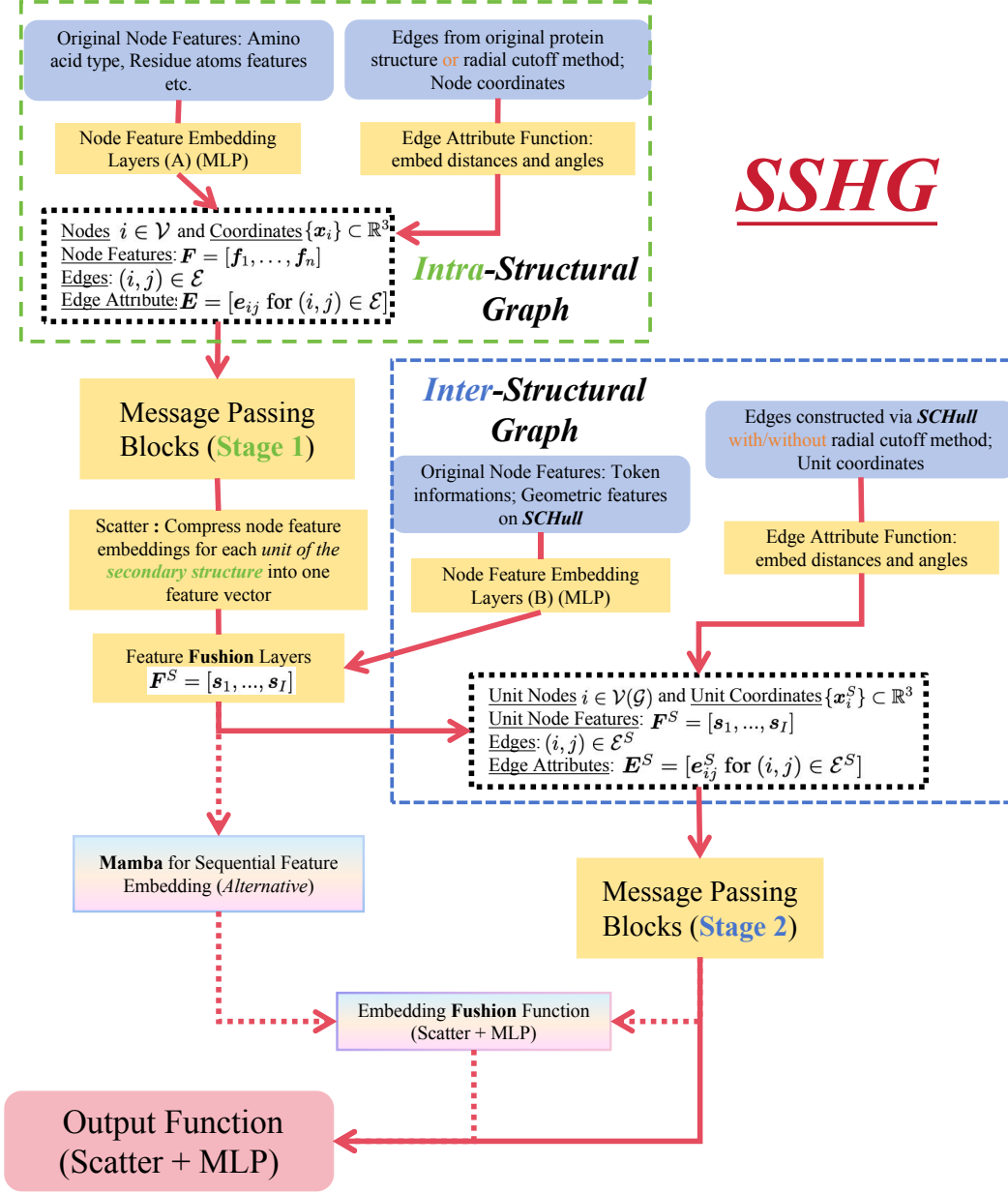


Figure 5: Illustration of the architectures of our SSHG model, with and without the integration of Mamba. The red arrows and red dashed arrows indicate the input-output dependencies in the SSHG and SSHG+Mamba models, respectively.

features for node i of the inter-structural graph as defined in equation 3. Then we discretise equation 10 into

$$\begin{aligned} h_{i+1} &= \bar{A}(x_i)h_i + \bar{B}(x_i)x_i \\ y_{i+1} &= \bar{C}(x_i)h_{i+1} \end{aligned} \quad (11)$$

and then obtain the outputs $s_i^{ssm} = y_{i+1}$ for $i \in \mathcal{V}(\mathcal{G})$.

Back to the message passing across structural units, we obtain the node features of the inter-structural graph $s_{global} = \text{readout}_2(\{\{s_i^{(T_2)} \mid i \in \mathcal{V}(\mathcal{G})\}\})$ in equation 3. Then we input s_i^{ssm} and s_{global} into an output function tailored for the task's target final outputs. Figure 5 shows the architecture of our model integrated with Mamba.

E.4 Discussion on Mamba

We observe that directly applying Mamba to the node sequences of the original protein graph increases the sequence length by over threefold, significantly slowing down the model. Despite this, the Mamba-only model achieves performance comparable to GNN-based methods such as ProNet, even without leveraging geometric information. This efficiency is largely due to `mamba_ssm`, a highly optimized CUDA C++ implementation that enables fast training of complex sequence models. In contrast, most GNN implementations are primarily in Python, which introduces overhead due to slower data processing and iterative computation.

To illustrate the difference, consider the typical pipeline structures:

- **GNN-based models (e.g., ProNet):**
 - Feature embedding → Python
 - Iteration over blocks → Python
 - Message passing → Python
 - Output layer → Python
- **Mamba_ssm-based models:**
 - Feature embedding → Python
 - Iteration over time steps → CUDA C++ (fused kernel)
 - Selective SSM operations → CUDA C++
 - Output layer → Python

The expensive recurrent and state-space computations in Mamba are fused into CUDA kernels, bypassing Python’s loop overhead. As a result, `mamba_ssm` narrows the performance gap without relying on structural priors. While it achieves results comparable to other baselines, we still observe significant improvements when incorporating SSHG.

E.4.1 Architecture and Experimental Setup

The number of message passing blocks, hidden channels, and dropout rates used for training SSHG on different tasks are listed in Table 6. The implementation of our methods is based on PyTorch and Pytorch Geometric, and all models are trained with the Adam optimizer. All are conducted on a single NVIDIA GeForce RTX 3090 24 GB. The hyperparameter searching space for training is shown in Table 7.

Hyperparameter	Values/Search Space	
	React	LBA
Number of layers (1st)	1, 2	1, 2
Number of layers (2nd)	2, 3	2, 3
Hidden channels	64, 128, 256	128, 192, 256
Dropout	0.2, 0.3, 0.5	0.2, 0.3
Mamba Blocks	4	4

Table 6: Model hyperparameters for SSHG

Hyperparameter	Values/Search Space	
	React	LBA
Epochs	500, 1000	300, 500
Batch size	16, 32	8, 16, 32
Learning rate	1e-4, 5e-4	5e-5, 1e-4, 2e-4
Learning rate scheduler	steplr	steplr
Learning rate decay factor	0.5	0.5
Learning rate decay epochs	50, 100	50, 100

Table 7: Training hyperparameters search space.

E.5 Additional Ablation Studies

	w/ SS	w/ hierarchical	w/ geometry	Test Acc
ProNet	\times	\times	\times	86.4
–	\checkmark	\times	\times	87.0
–	–	\checkmark	\times	87.2
–	–	\checkmark	\checkmark	87.5
GVPGNN[18]	\times	\times	\times	68.5
–	\checkmark	\times	\times	66.7
–	–	\checkmark	\times	71.5
–	–	\checkmark	\checkmark	73.6

Table 8: **Feature Selection:** Different GNNs with (w/) or without (w/o) hierarchical geometric graphs, geometric features $g_i^\top g_j$, or secondary structure tokens.

Table 8 shows that simply appending SS tokens as a feature to the original GNNs does not necessarily improve performance. In contrast, combining SS information and geometric features through a dedicated hierarchical mechanism leads to consistent improvements.