SUMMARY OF THE APPENDIX

This appendix provides more details of our approach, additional literature review, further discussions, additional experimental results, broader impacts, and limitations. These topics are organized as follows:

- §A: Details of Training Setup
- §B: Details of Evaluation Metrics
- §C: Rotation Invariance
- §D: Clustering Algorithm
- §E: Additional Literature Review
- §F: More Quantitative Results
- §G: More Qualitative Results
- §H: Broader Impacts
- §I: Limitations

A. Details of Training Setup

In our experiments, we train our framework using a SGD optimizer with learning rate of 1e-3 and a weight decay of 5e-4. Due to memory constraints, we set the batch size to 32, 24, 8, and 8 for EC, GO, Fold Classification, and Reaction tasks, respectively. The framework comprises four iterations of clustering, which employ varying numbers of channels at each iteration. For Fold Classification, we use 256, 512, 1024 and 2048 channels for the four iterations, respectively. For EC, GO and Reaction, we use 128, 256, 512 and 1024 channels for the four iterations, respectively. The training process spans 200 or 300 epochs for each dataset, and the best model is selected based on the validation performance. Details can be seen in Table 7.

In addition, we adopt data augmentation techniques, similar to those used in [22, 38] to augment the data for fold and reaction classification tasks. Specifically, we apply Gaussian noise with a standard deviation of 0.1 and anisotropic scaling within the range of [0.9, 1.1] to the amino acid coordinates in the input data. We also add the same noise to the atomic coordinates within the same amino acid to ensure that the internal structure of each amino acid remains unchanged.

B. Details of Evaluation Metrics

We first present the details of evaluation metrics for enzyme commission number prediction and gene ontology term prediction. The objective of these tasks is to determine whether a protein possesses specific functions, which can be viewed as multiple binary classification tasks. We define the first metric as the protein-centric maximum F-score ($F_{\max}$). This score is obtained by calculating the precision and recall for each protein and then averaging the scores over all proteins. To be more specific, for a given target protein i and a decision threshold $\lambda \in [0, 1]$, we compute the precision and recall as follows:

$$\text{precision}_i(\lambda) = \frac{\sum_a \mathbb{1}[a \in P_i(\lambda) \cap G_i]}{\sum_a \mathbb{1}[a \in P_i(\lambda)]},$$

$$\text{recall}_i(\lambda) = \frac{\sum_a \mathbb{1}[a \in P_i(\lambda) \cap G_i]}{\sum_a \mathbb{1}[a \in G_i]},$$

where a represents a function term in the ontology, G_i is a set of experimentally determined function terms for protein i, $P_i(\lambda)$ denotes the set of predicted terms for protein i with scores greater than or equal to λ, and $\mathbb{1}[\cdot] \in \{0, 1\}$ is an indicator function that is equal to 1 if the condition is true.

Then, the average precision and recall over all proteins at threshold λ is defined as:

$$\text{precision}(\lambda) = \frac{\sum_i \text{precision}_i(\lambda)}{M(\lambda)},$$

$$\text{recall}(\lambda) = \frac{\sum_i \text{recall}_i(\lambda)}{N},$$

where we use N to represent the number of proteins, and $M(\lambda)$ to denote the number of proteins on which at least one prediction was made above threshold λ, *i.e.*, $|P_i(\lambda)| > 0$. By combining these two measures, the maximum F-score is defined as the maximum F-measure value obtained across all thresholds:

$$F_{\max} = \max_{\lambda} \left\{ \frac{2 \times \text{precision}(\lambda) \times \text{recall}(\lambda)}{\text{precision}(\lambda) + \text{recall}(\lambda)} \right\}.$$

The second metric, mean accuracy, is calculated as the average precision scores for all protein-function pairs, which is equivalent to the micro average precision score for multiple binary classification.

C. Rotation Invariance

Rotation Invariance. To make our method rotationally invariant [71], we augment the distance information d_k^n by using a relative spatial encoding [40]:

$$d_k^n = \left(d(||\boldsymbol{z}_k - \boldsymbol{z}_n||),\ \boldsymbol{O}_n^\top \frac{\boldsymbol{z}_k - \boldsymbol{z}_n}{||\boldsymbol{z}_k - \boldsymbol{z}_n||},\ q(\boldsymbol{O}_n^\top \boldsymbol{O}_k^n)\right), \quad (6)$$

where $\boldsymbol{O}_n = [\boldsymbol{b}_n,\ \boldsymbol{j}_n,\ \boldsymbol{b}_n \times \boldsymbol{j}_n]$, $\boldsymbol{b}_n$ denotes the negative bisector of angle between the ray $(v_{n-1} - v_n)$ and $(v_{n+1} - v_n)$, and $\boldsymbol{j}_n$ is a unit vector normal to that plane. Formally, we have $\boldsymbol{u}_n = \frac{\boldsymbol{z}_n - \boldsymbol{z}_{n-1}}{||\boldsymbol{z}_n - \boldsymbol{z}_{n-1}||} \in \mathbb{R}^3$, $\boldsymbol{b}_n = \frac{\boldsymbol{u}_n - \boldsymbol{u}_{n+1}}{||\boldsymbol{u}_n - \boldsymbol{u}_{n+1}||} \in \mathbb{R}^3$, $\boldsymbol{j}_n = \frac{\boldsymbol{u}_n \times \boldsymbol{u}_{n+1}}{||\boldsymbol{u}_n \times \boldsymbol{u}_{n+1}||} \in \mathbb{R}^3$, where $\times$ is the cross product. The first term in Eq. 6 is a distance encoding $d(\cdot)$ lifted into the radius r, the second term is a direction encoding that corresponds to the relative direction of $v_k^n \to v_n$, and the third term is an orientation encoding $q(\cdot)$ of the quaternion representation of the spatial rotation matrix. This encoding approach allows us to capture both local and global geometric information while being invariant to different orientations. Related experimental results are seen in Table 8.

D. Neural Clustering Algorithm

Table 7. The hyperparameter configurations of our method vary across different tasks. We choose all the hyperparameters based on their performance on the validation set. See details in §A.

Hyperparameter	EC	GO-BP	GO-MF	GO-CC	Fold Classification	Reaction
batch size	32	24	24	24	8	8
Channels	[128,256,512,1024]	[128,256,512,1024]	[128,256,512,1024]	[128,256,512,1024]	[256,512,1024,2048]	[128,256,512,1024]
# epoch	300	300	300	300	200	200

Algorithm 1: Pseudo-code of Neural Clustering

Input : Protein $\mathcal{P} = (\mathcal{V}, \mathcal{E}, \mathcal{Y})$; Amino acid embedding $\boldsymbol{e}_j$ for amino acid $v_j \in \mathcal{V}$; Cluster nomination ratio ω; Nomination operator NOMINATE; Index selection operator INDEXSELECT; Add self-loop operator ADDSELFLOOPS; Spherosome clustering operator RADIUS; Spherosome clustering radius r; ReLU activation function σ; Geometric coordinates Pos; Geometric orientations Ori; Sequential orders Seq

Intermediate: Clustered features X^c and Scored cluster features $\hat{X}^c$; Adjacency matrix A; Edge index E; Cluster scores vector Φ; Nominated index $index$

Output : Nominated amino acid features X, coordinates Pos, orientations Ori, sequential orders Seq

for $t = 1, 2, 3, 4$ **do**
  $A \leftarrow \text{RADIUS}(Pos, r)$;
  $E \leftarrow \text{ADDSELFLOOPS}(A)$;
  for $n = 1...N_{t-1}$ **do**
    $\tilde{\boldsymbol{x}}_n \leftarrow \vec{0}$;
    for $k = 1...K$ **do**
      $\boldsymbol{g}_k^n, \boldsymbol{o}_k^n, d_k^n, s_k \leftarrow (Pos, Ori, Seq)$;
      $\boldsymbol{x}_k^n \leftarrow f(\boldsymbol{g}_k^n, \boldsymbol{o}_k^n, d_k^n, s_k, \boldsymbol{e}_k)$;
      $\gamma_k^n \leftarrow softmax(\sigma([\boldsymbol{W}_1 \boldsymbol{x}_n, \boldsymbol{x}_k^n]))$;
      $\tilde{\boldsymbol{x}}_n \leftarrow \tilde{\boldsymbol{x}}_n + \gamma_k^n \boldsymbol{x}_k^n$;
    end
    $X_n^c \leftarrow \tilde{\boldsymbol{x}}_n$;
  end
  $\Phi \leftarrow \text{GCN}(X^c, E)$;
  $\hat{X}^c \leftarrow \Phi \odot X^c$;
  $N_t \leftarrow \lfloor \omega \cdot N_{t-1} \rfloor$;
  $index \leftarrow \text{NOMINATE}(\Phi, N_t)$;
  $X, Pos, Ori, Seq \leftarrow$ INDEXSELECT$(\hat{X}^c, Pos, Ori, Seq, index)$;
end

E. Additional Literature Review

Graph Pooling. Graph pooling designs have been proposed to achieve a useful and rational graph representation. These designs can be broadly categorized into two types [53]: Flat Pooling [6, 19, 84, 88, 93] and Hierarchical Pooling [3, 27–29, 50, 57, 91]. Flat Pooling generates a graph-level representation in a single step by primarily calculating the average or sum of all node embeddings without consideration of the intrinsic hierarchical structures of graphs, which causes information loss [47]. On the other hand, Hierarchical Pooling gradually reduces the size of the graph. Previous graph pooling algorithms, as variants of GCN, still follow the message passing pipeline. Typically, they are hard to be trained and need many extra regularizations and/or operations. For example, Diffpool [91] is trained with an auxiliary link prediction objective. Besides, it generates a dense assignment matrix thus incurring a quadratic storage complexity. Top-K pooling [27], adopts a Unet-like, graph encoder-decoder architecture, which is much more complicated than our model but only learns a simple scalar projection score for each node.

In contrast, our clustering-based algorithm is more principled and elegant. It can address the sparsity concerns of Diffpool and capture rich protein structure information by aggregating amino acids to form clusters instead of learning from a single node. Furthermore, it sticks to the principle of clustering throughout its algorithmic design: SCI step is to form the clusters by considering geometrical relations among amino acids; CRE step aims to extract cluster-level representations; CN step is for the selection of important cluster centers. It essentially combines unsupervised clustering with supervised classification. The forward process of our model is inherently a neural clustering process, which is more transparent and without any extra supervision.

F. More Quantitative Results

Rotation Invariance. We compare our framework with and without rotation invariance on all four tasks. The results are shown in Table 8, where we can see that rotation invariance improves the performance of our framework on all tasks. For example, on protein fold classification, rotation invariance boosts the average accuracy from 76.4% to 81.3%. This indicates that rotation invariance can help our framework to capture the geometric information of proteins more effectively and robustly.

Table 8. Analysis of the impact of rotation invariance and different numbers of CRE blocks (§4.5).

Rotation Invariant	B	EC	GO			Fold Classification				Reaction
			BP	MF	CC	Fold	Super.	Fam.	Avg.	
✔	*1*	0.825	0.430	0.618	0.464	57.7	76.3	99.4	77.8	87.6
✔	*2*	**0.866**	**0.474**	**0.675**	**0.483**	**63.1**	**81.2**	**99.6**	**81.3**	**89.6**
✔	*3*	0.857	0.466	0.669	0.474	61.8	80.2	99.5	80.5	88.9
✘	*2*	0.781	0.392	0.614	0.436	56.4	75.3	97.9	76.4	87.1

Number of CRE Blocks. In our framework, we use B CRE blocks at each clustering iteration to extract cluster features. We study the impact of using different values of B from 1 to 3 on all four sub-tasks. We stop using $B > 3$ as the required memory exceeds the computational limit of our hardware. The results are shown in Table 8, where we can find that $B = 2$ achieves the best performance on all tasks. For instance, on enzyme reaction classification, $B = 2$ achieves an accuracy of 89.6%, while if $B = 1$ or $B = 3$, the accuracy drops to 87.6% and 88.9%, respectively. This suggests that using two CRE blocks is sufficient to capture the cluster information and adding more blocks does not bring significant improvement but increases the computational cost.

G. More Qualitative Results

More visualization results of the clustering results are presented in Fig. 7. The color of the node denotes the score calculated in our CN step. For example, in the first row, we can see that the protein '1rco.E' has a helical structure with some loops. After the first iteration of clustering, our method selects some amino acids that are located at the ends or bends of these loops and helices as the center nodes for the next iteration. These amino acids may play an important role in stabilizing the protein structure or interacting with other molecules. After the second iteration of clustering, our method further narrows down the number of center nodes by selecting those that have high scores. These amino acids may form functional domains or motifs that are essential for the protein function. Our method finally identifies a few amino acids that have the highest scores and are most representative of the protein structure and function. Through visualizing the clustering results at each iteration, we can explicitly understand how our method progressively discovers the critical components of different proteins by capturing their structural features in a hierarchical way.

In addition, as shown in the figure, we present pairs of protein chains from the same family or same protein (*i.e.*, '1rco.R' and '1rco.E', '3n3y.B' and '3n3y.C', '6gk9.B' and '6gk9.D'). For example, we observe that the clustering results of '3n3y.B' and '3n3y.C' (two chains of the same protein) are very similar, indicating that they have similar critical amino acids that determine their structure and function. This observation is consistent with the biological reality that proteins from the same family or same protein often have long stretches of similar amino acid sequences within their primary structure, suggesting that our method is effective in identifying these critical amino acids.

H. Broader Impacts

Our neural clustering framework for protein representation learning has several potential applications and implications for society. Protein representation learning can help advance our understanding of protein structure and function, which are essential for many biological processes and diseases. By discovering the critical components of proteins, our method can inspire protein design, engineering, and modification, which can lead to the development of novel therapies, drugs, and biotechnologies. For example, our framework can assist in designing new protein sequences that exhibit specific properties or functions, such as catalyzing biochemical reactions or binding to other molecules. This can enable the creation of new enzymes, antibodies, vaccines, and biosensors that can have a positive impact on human health and well-being.

I. Limitations

Our neural clustering framework for protein representation learning has several limitations that need to be addressed in future work. First, our framework relies on the availability of protein structures, which are not always easy to obtain or predict. Although our framework can leverage both sequence-based and structure-based features, it may lose some information that is only encoded in the 3D structure. Second, our framework assumes that the critical components of a protein are determined by its amino acid sequence and structure, but it does not consider other factors that may affect protein function, such as post-translational modifications, interactions with other molecules, or environmental conditions. Third, our framework does not explicitly account for the evolutionary relationships among proteins, which may provide useful information for protein representation learning. Incorporating phylogenetic information into our framework may enhance its ability to capture the functional diversity and similarity of proteins.

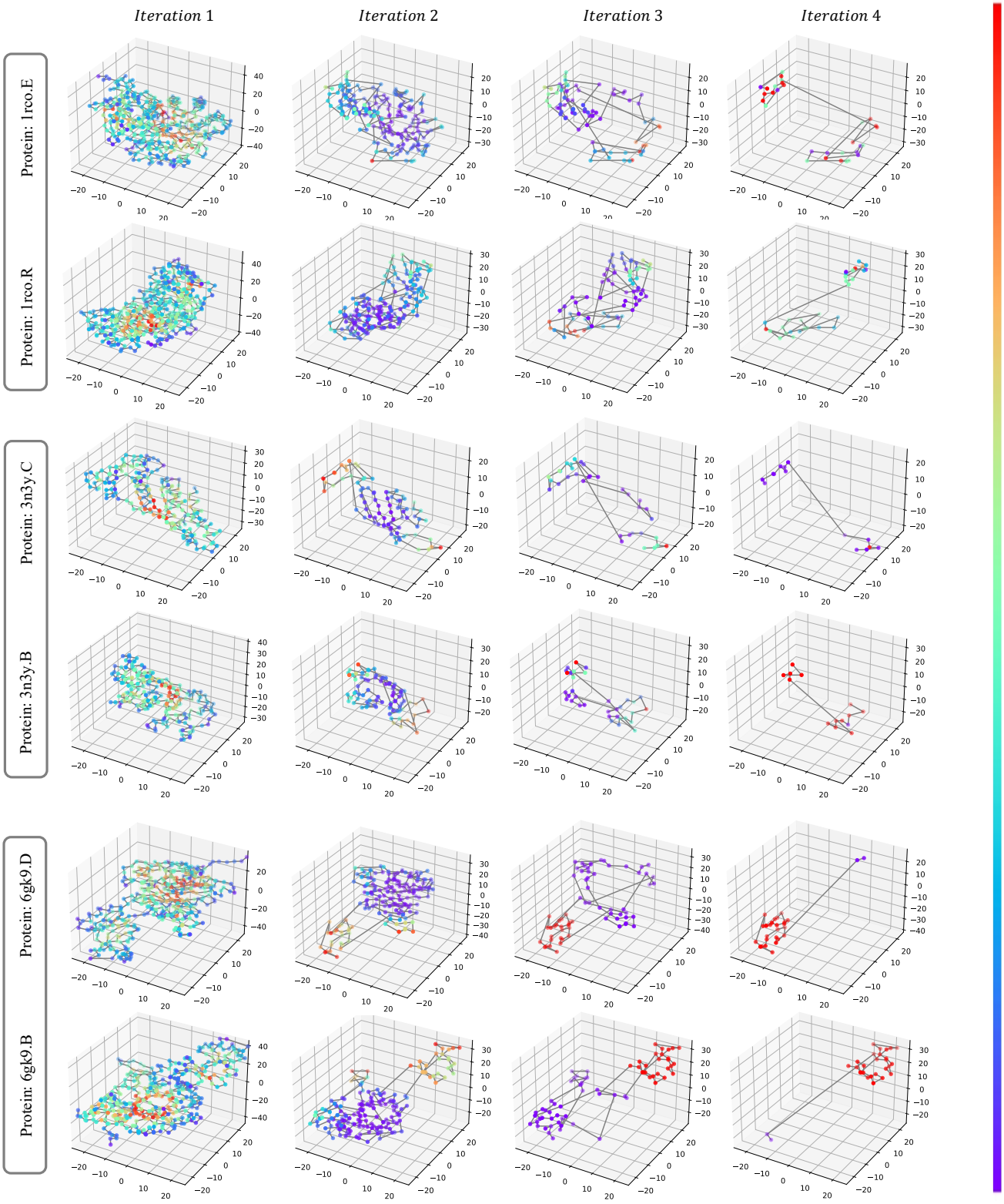

Figure 7. More visualization results. See related analysis in §G.