A Variational Graph Partitioning Approach to Modeling Protein Liquid-liquid Phase Separation

Gaoyuan Wang[1,2†] **Jonathan H Warrell** [1,2†] **Suchen Zheng**[1,2] **Mark Gerstein**[1,2,3,4,5*]

[1] Program in Computational Biology and Bioinformatics,
[2] Department of Molecular Biophysics and Biochemistry,
[3] Department of Computer Science,
[4] Department of Statistics & Data Science,
[5] Department of Biomedical Informatics & Data Science,
Yale University, New Haven, Connecticut 06520, USA
† These authors contributed equally to this work.
*Corresponding author: mark@gersteinlab.org

Abstract

Protein Liquid-Liquid Phase Separation (LLPS) plays an essential role in cellular processes and is known to be associated with various diseases. However, our understanding of this enigmatic phenomena remains limited. In this work, we propose a graph-neural-network(GNN)-based interpretable machine learning approach to study the intricate nature of protein structure-function relationships associated with LLPS. For many protein properties of interest, information relevant to the property is expected to be confined to local domains. For LLPS proteins, the presence of intrinsically disordered regions (IDR)s in the molecule is arguably the most important information; an adaptive GNN model which preferentially shares information within such units and avoids mixing in information from other parts of the molecule may thus enhance the prediction of LLPS proteins. To allow for the accentuation of domain restricted information, we propose a novel graph-based model with the ability to partition each protein graph into task-dependent subgraphs. Such a model is designed not only to achieve better predictive performance but also to be highly interpretable, and thus have the ability to suggest novel biological insights. In addition to achieving state-of-the-art results on the prediction of LLPS proteins from protein structure for both regulator and scaffold proteins, we examine the properties of the graph partitions identified by our model, showing these to be consistent with the annotated IDRs believed to be largely responsible for LLPS. Moreover, our method is designed in a generic way such that it can be applied to other graph-based predictive tasks with minimal adaption.

1 Introduction

LLPS [1, 2, 3, 4] is a phenomena underlying vital biological processes, such as the formation of membrane-less organelles in cells. Unlike membrane-bound organelles, where lipid bilayer membranes are present to encapsulate molecules spatially, LLPS describes the case where multivalent interactions between protein chains cause the formation of a distinct phase of protein condensates. Recent studies revealed evidences of the association of LLPS with the development of human diseases, such as neurodegenerative diseases and cancer [2, 5, 6, 7].

The ability of proteins to form condensates through LLPS is believed to be largely driven by the enrichment of certain binding domains in the molecule, also called IDRs [8, 9]. It is widely accepted that cation-pi, pi-pi [10], electrostatic, hydrophobic and hydrogen bonds interactions play an important role in mediating LLPS of IDRs. Yet, a quantitative understanding of LLPS driving forces is still missing. Here, we are interested in protein structure-function prediction for modeling protein LLPS. However, the large variety of distinct IDR sequence

features [11, 12] and other possible unknown driving forces of LLPS makes the task difficult for conventional physical modeling approaches, but a suitable target for machine learning tools. Having an interpretable predictive tool to distinguish proteins based on their ability to phase separate will not only help us to identify phase separating proteins but also to find the key domains and properties responsible for phase separation.

In many domain areas, Graph Neural Network (GNN)s have emerged as a powerful tool for representation learning of systems with rich-relational data. These include social networks[13], traffic networks[14] and biological networks [15, 16, 17]. Using GNNs to study proteins, where protein structures are for instance encoded as graphs and node features carry amino acid properties, is a promising approach which has already been shown to outperform sequence-based convolutional neural networks in different tasks [18, 19, 20].

In general, the identification and choice of the optimal underlying graph for a task is crucially important for achieving high performance with GNN. For instance, it has been observed that sparsification [21] and graph structure learning [22] can enhance performance, and that such methods can avoid the problem of 'oversmoothing' observed in many GNN models[22]. In particular for proteins, we expect protein function relevant information to be confined to local subgraphs, supported by the fact that many protein functions are carried-out by domains. Specifically for LLPS prediction, it may be beneficial to partition the molecule graph, and remove connecting edges between task-irrelevant regions and IDRs, thereby preventing message passing from task-irrelevant regions to the IDRs. Combining the imperative to incorporate sparsification mechanisms to mitigate against oversmoothing, and the potential benefit of emphasizing distinct protein functional domains, we are motivated to propose a novel GNN architecture, with the ability to learn advantageous graph partitions on-the-fly, and hence learn improved graph representations. In our method, prior to the graph representation learning through message passing, edges between learned subgraphs are removed (sparsification) in an adaptive fashion, to ensure that information sharing is confined to local subgraphs in the initial layers of the network.

Previous approaches to joint graph structure and representation learning in GNNs have used graph encoders [23] and alternating optimization [24] methods. Also related to our approach are methods such as DropEdge [21] and other sparsification approaches [25], which aim to remove task-irrelevant edges, and attention-based approaches [26], which may be viewed as soft structure learning methods. Methods such as [21] learn independent measures of edge importance, and hence cannot enforce global constraints on the predicted graph structures. Further, although generative models such as [23] and [25] can in principle learn such constraints, they do not provide a mechanism for imposing such constraints in advance, and only enforce restricted forms of sparsity constraints (such as k-neighborhood subgraphs [25]). Likewise, attention-based methods do not enforce global constraints or sparsity directly. In contrast to the existing methods mentioned above, our approach allows desired global characteristics to be imposed onto the resulting subgraphs that emerge.

Our architecture starts with a layer of trainable meta-features, which determine the graph structure of subsequent layers; these meta-feature vectors are combined with graph-based features as input to an arbitrary clustering algorithm in order to capture important protein domains. Then, subsequent GNN layers predict phase separation ability using the combined domain characteristics.

Optimization of our model in an end-to-end manner is challenging, due to the non-differentiability of the clustering layer. A wide range of black-box optimization techniques [27] has been developed to handle objective functions which may not be globally differentiable. Promising recent approaches include score-function methods with control variates [27], variational optimization (VO) [28], and scale-space-based continuation approaches [29]. Such approaches have been applied in the context of reinforcement learning [27], graph matching [30], hyperparameter optimization [31] and differentiable deep neural network training [28], but there has been limited application of such techniques specifically to GNN architectures. In this work, we introduce an efficient VO approach, based on optimization of an smoothed objective function using a Gaussian scale-space [32]. This approach allows us to derive an enhanced form of the smoothing-based optimization algorithm [30] that makes use of local

gradient information, which is important for efficiently training the graph-based layers of our model, while not assuming the objective is globally differentiable.

Our contributions may be summarized in four key areas as follows: First, we introduce a novel GNN model adept at addressing predictive tasks, such as protein LLPS prediction, where unknown subsystems may be crucial for information extraction. Second, we present a novel metric for assessing structural-functional similarity in protein regions in eq. (3) that can be adapted to different tasks. Third, combining the above, we develop a state-of-the-art model for LLPS protein prediction, where message passing is confined to learned subregions of proteins; our model is shown to outperform all prior methods for LLPS prediction. Lastly, we corroborate the biological significance of the protein regions identified by our model by comparing them with annotated protein disordered regions.

We first outline our novel GNN architecture with cluster-based constraints and our VO framework in Secs. 2.1 and 2.2 respectively. Sec. 3 demonstrates the effectiveness of our model in predicting protein LLPS on both regulator and scaffold proteins, in addition to analysing the properties of the graphs learned by our model and validating the identified protein subregions against external biological annotations. Sec. 5 concludes with a discussion, and further results are given in Supplementary section S1, where we apply our approach to synthetic graph classification tasks with known ground truth graph structures.

2 Methods

Our overall approach is illustrated in fig. 1 and summarized in the following. Detailed descriptions of our novel graph architecture and VO approach are given in sec. 2.1 and sec. 2.2 respectively.

1. Each protein in the dataset is encoded by a graph $\mathcal{G} = \{\mathbf{X}, E\}$, where node i carries feature vector $\mathbf{x}_i$.
2. Initialize the variational distribution $Q(\theta)$.
   - 2a) The *Pre-Linear* layer's weights W^{meta} are drawn from the variational distribution $Q(\theta)$. Applying the *Pre-Linear* layer yields a meta-feature vector for each data-point, $\mathbf{m}_i = W^{\text{meta}}\mathbf{x}_i^0$, which form the rows of the meta-feature matrix, $\mathbf{M}$.
   - 2b) For each node pair i and j, a similarity score is calculated as
   $$S_{ij} = -\{|\mathbf{m}_i - \mathbf{m}_j| + C_1 \times \mathit{distance}(i,j) + C_2 \times |\mathit{degree}(i) - \mathit{degree}(j)|\}$$
   using information stored in $\mathbf{M}$.
   - 2c) Using Affinity Propagation with the similarity matrix obtained from S_{ij}, each protein graph is clustered into subgraphs. Edges between different subgraphs are removed, resulting in a fractionized graph $\mathcal{G}^F = \{\mathbf{X}, E^F\}$.
   - 2d) A conventional GNN is trained on $\mathcal{G}^F$ and the negative CrossEntropyLoss serves as the performance score for each W^{meta} sample, corresponding to a specific graph partition.
   - 2e) Update the variational distribution $Q(\theta)$ via Smoothing-based optimization (SBO) based on the performance scores. And repeat 2a) to 2d) until convergence is attained.
3. The final prediction is made using the optimal graph partition and the optimized GNN.

2.1 Adaptive Cluster-based Architecture

We assume that our data consists of graphs of the form $\mathcal{G} = \{\mathbf{X}, E\}$, where the graph structure is given by the preliminary set of edges $E \subset N \times N$, where N is the set of nodes, and $\mathbf{X}$ denotes the matrix of D-dimensional features associated with the N nodes in the graph, hence for node i we denote its associated feature vector as $\mathbf{x}_i \in \mathbb{R}^D$. Additionally, for each data-point, we have an associated binary label $Y \in \{0, 1\}$.

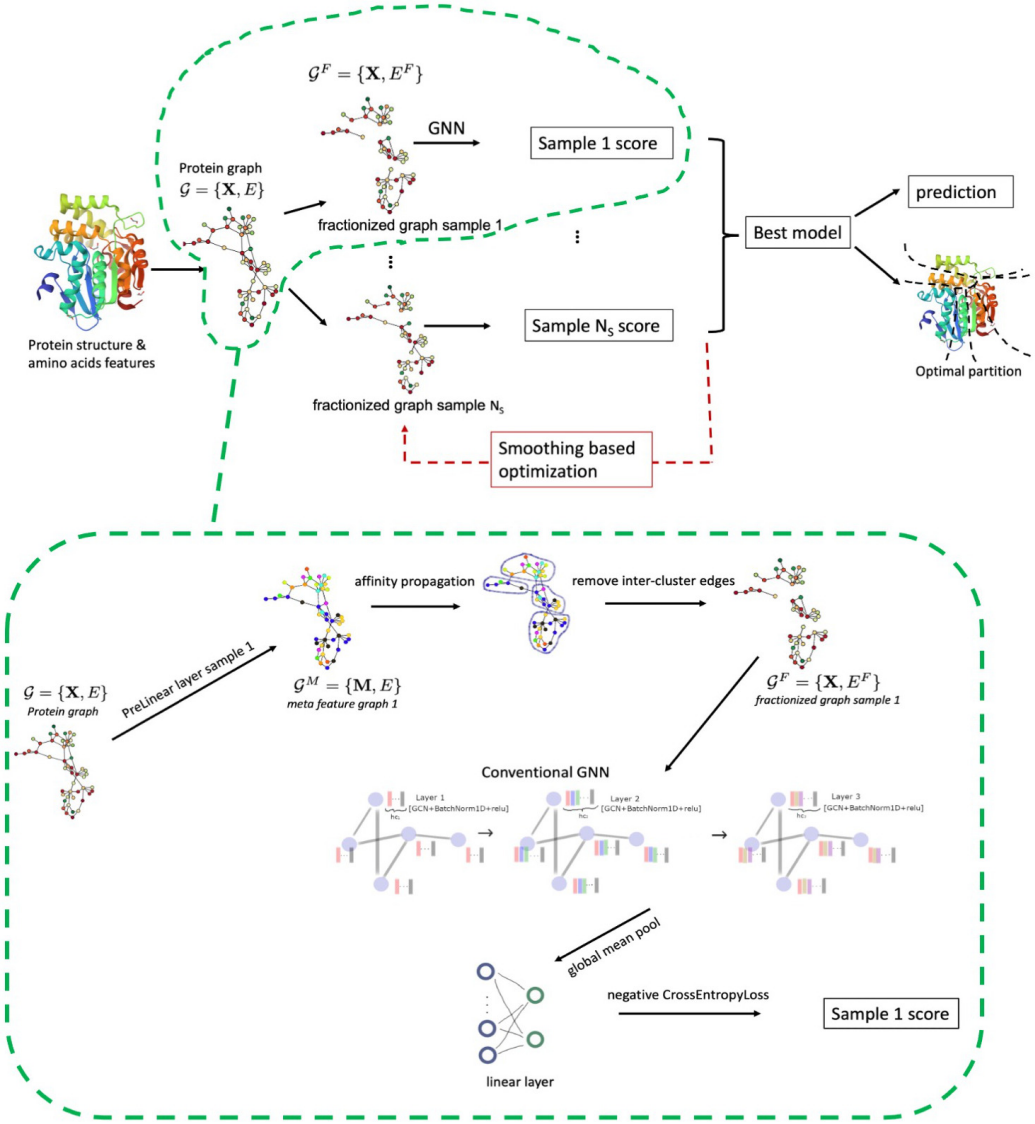

Figure 1: The general GNN architecture with adaptive cluster layers illustrated. The lower green dashed box provides a more detailed rendition of the steps for one sample within one SBO meta-epoch outlined in the upper green dashed box. For meta-epoch t, N_S samples of *Pre-Linear* layer (PLL) parameters are drawn from a multivariate normal distribution with mean μ_t and covariant matrix $(\sigma_t)^2\mathbf{I}$. For each sample, we perform affinity clustering to obtain an unique set of refined graphs. We train a GNN model on the refined graphs and use the training score of the epoch with the best validation accuracy as the score function of this sample to update μ and σ using eq. (7).

We outline our architecture below assuming GCNConv-based message-passing [33] for convenience, although in the experimentation we also consider TransformerConv [34] message-passing formulations. Here, we let L denote the number of layers, while $\mathbf{x}_i^l \in \mathbb{R}^{D'}$ is the node feature vector of node i in layer l, and W^l is the layer-specific trainable weight matrix. In addition, we assume we have a further set of weights $W^{\text{meta}} \in \mathbb{R}^{D\times D}$ which we use to generate a vector of 'meta-features' $\mathbf{m}_i$ associated with each node, and a clustering algorithm $\mathcal{A} : \mathbb{R}^{N\times D} \to \mathcal{K}^N$, which returns a cluster index ($\mathcal{K} = \{1...N_K\}$) for each node depending on the input collection of meta-features. Finally, $\mathcal{N}(i, \mathbf{M})$ returns the neighbours of node i in our model, which here include only those points connected to i by an edge in E belonging to

the same cluster as determined by the matrix of meta-features $\mathbf{M}$. Our model can thus be summarized as:

$$\begin{aligned} \mathbf{m}_i &= W^{\text{meta}}\mathbf{x}_i^0, \qquad \mathbf{x}_i^{l+1} = \text{RELU}(W^l \sum_{j\in\mathcal{N}(i,\mathbf{M})\cup i} \frac{e_{ij}}{\sqrt{d_j d_i}}\mathbf{x}_j^l) \\ \mathcal{N}(i,\mathbf{M}) &= \{j \in N \mid i,j \in E,\ K_i = K_j,\ \mathcal{A}(\mathbf{M}) = K\} \end{aligned} \tag{1}$$

with $d_i = 1 + \sum_{j\in\mathcal{N}(i)}$, $\mathbf{X}^0 = \mathbf{X}$, and $\text{RELU}(a) = \max(a, 0)$. We refer to W^{meta} as a '*Pre-Linear*' layer, since it is a linear transform of the input features. Also, as noted below, we apply batch correction to each layer, which simply applies a linear transformation to each W^l matrix. Finally, we predict the model output by pooling the final node features and multiplying by a dense output layer, $W^{\text{final}} \in \mathbb{R}^{D\times 1}$:

$$P(Y = 1|G) = \sigma(\frac{1}{N}W^{\text{final}}\sum_i \mathbf{x}_i^L) \tag{2}$$

where σ is the sigmoid function, $\sigma(a) = 1/(1 + \exp(-a))$.

Clustering Algorithm: In this work, we use affinity propagation (AP) as our clustering algorithm $\mathcal{A}$ [35]. Using meta-feature vectors $\mathbf{m}_i$ and the preliminary graph topology defined by E, we calculate a similarity matrix for each graph. We propose to use the following definition of similarity to capture how likely node i and j belong to the same cluster:

$$S_{ij} = -\{|\mathbf{m}_i - \mathbf{m}_j| + C_1 \times \textit{distance}(i,j) + C_2 \times |\textit{degree}(i) - \textit{degree}(j)|\}. \tag{3}$$

The shortest-path distance between two nodes $\textit{distance}(i,j)$ is defined as the number of edges in E along the shortest-path between i and j. $\textit{degree}(i)$ is the degree of node i defined as the number of edges in E incident this node. C_1 and C_2 are two scaling factors regulating the relative importance of each term. $\mathcal{A}(\mathbf{M})$ is then set to the clustering returned by the AP algorithm, using similarity matrix S as input.

2.2 Smoothing-based Variational Optimization Framework

We assume we have a dataset $\mathcal{D}$ consisting of pairs (G, Y) of matching graphs and labels. During training, our goal is thus to optimize the log-probability of our dataset: $F(\theta) = \sum_{(G,Y)\in\mathcal{D}} \log P(Y|G,\theta)$, where $\theta = \{W^{1\ldots L}, W^{\text{meta}}, W^{\text{final}}\}$. Since F includes discontinuities due to the clustering algorithm $\mathcal{A}$, we introduce a variational distribution $Q(\theta) = \mathcal{N}(\theta|\mu, \sigma^2\mathbf{I})$, with $\mathbf{I}$ the identity matrix, where $\mathcal{N}(.|\mu, \Sigma)$ is a multivariate Gaussian distribution, and an associated smoothed variational objective:

$$\mathcal{L}_i^{smooth}(\mu, \sigma) = \mathbb{E}_Q[F(\theta)]. \tag{4}$$

By optimizing $\mathcal{L}^{smooth}$ we are optimizing a lower-bound on $\mathcal{L}$, since $\mathbb{E}_Q[F(\theta)] \leq \max_\theta F(\theta)$.

Since F is continuous within local regions (i.e. for a fixed W^{meta}), we can further strengthen the smoothing objective above using local optimization. We assume we have an effective local optimizer $\mathcal{O}$ for which $\mathcal{O}(\theta) = \theta'$ such that $F(\theta') \geq F(\theta)$. Then we can define an augmented variational objective as:

$$\mathcal{L}_i^{aug}(\mu, \sigma) = \mathbb{E}_Q[F'(\theta)], \quad F'(\theta) = F(\mathcal{O}(\theta)), \tag{5}$$

which provides a tighter bound on the original loss, since:

$$\mathbb{E}_Q[F(\theta)] \leq \mathbb{E}_Q[F'(\theta)] \leq \max_\theta F(\theta). \tag{6}$$

To optimize $\mathcal{L}_i^{aug}$, we use smoothing-based optimization (SBO) [30], combined with local gradient descent. Hence, we begin by fixing the variational distribution, $\{\mu_t, \sigma_t\}$ at $t = 0$, and draw samples $\theta_{s=1\ldots N_S}$ from $\mathcal{N}(.|\mu_t, \sigma_t^2\mathbf{I})$. We then apply our local gradient descent optimizer $\mathcal{O}$ to each sample, which fixes the meta-features while optimizing the subsequent layers of the GNN, generating $\theta'_{s=1\ldots N_S}$. We calculate $f_s = \max(F'(\theta_s) + c, 0) = \max(F(\theta'_s) + c, 0)$, where c is an offset to ensure that f_s is non-negative, and calculate μ_{t+1} and σ_{t+1} using the following updates:

$$\mu_{t+1} = \frac{\sum_s f_s\theta_s^t}{\sum_s f_s} \qquad \text{and} \qquad \sigma_{t+1} = \sqrt{\frac{\sum_s f_s|\theta_s^t - \mu_t|_2^2}{N_S\sum_s f_s}}. \tag{7}$$

Algorithm 1 Smoothing-based Optimization Augmented with Local Gradients

Input $\mathcal{D} = \{G, Y\}, \mu_0, \sigma_0, \mathcal{A}, N_S, T, c$
Initialize $\mu = \mu_0$, $\sigma = \sigma_0$
for $t = 1 \dots T$ **do**
  Draw weight matrices $W^{\text{meta},(s)}$ for samples $1 \dots N_S$ from $\mathcal{N}(.|\mu, \sigma^2\mathbf{I})$
  Calculate meta-features $\mathbf{M}^{(s,d)} = W^{\text{meta},(s)}\mathbf{X}^{0,(s,d)}$ for each sample s, data-point d
  for $s = 1 \dots N_S$ **do**
    Initialize $W^{1\dots\text{L},(s)}$, $W^{\text{final},(s)}$ using a standard normal distribution
    Calculate clustering for each data-point d using $K^{(d)} = \mathcal{A}(\mathbf{M}^{(s,d)})$
    Optimize $W^{1\dots L,(s)}$, $W^{\text{final},(s)}$ via stochastic gradient descent, using fixed sparse graphs
      determined by the clusterings $K^{(d)}$ (eq. 1)
    Set f_s to $\max(\log\Big(P(Y|W^{(s)}, G)\Big) + c, 0)$
  end for
  Update μ and σ using eq. (7), using the weights f_s
end for
Output μ_t, σ_t, $W^{(s,t)}$, $f_{s,t}$ for $t = 1 \dots T$, $s = 1 \dots N_S$

We note that, although the updates in eq. (7) may be applied simultaneously as in [30], we found that applying them sequentially improved the convergence of the algorithm to a local optimum. Also, in practice we use the default PyTorch-Geometric parameter initialization settings for $W^{1\dots\text{L}}$ and W^{final}, and restrict the distribution Q to the parameters W^{meta}; this is equivalent to setting $\mu_i = 0$ for all other parameters, and using two separate standard deviations, σ_a and σ_b for W^{meta} and all other parameters respectively, where only σ_a is updated according to eq. (7), and the second remains fixed at $\sigma_b = 1$ (hence all other parameters are always initialized using a standard normal distribution). We summarize our smoothing-based local stochastic gradient descent algorithm in Alg. 1.

3 Results

We apply our model to two LLPS tasks, involving predicting protein LLPS for regulator and scaffold proteins. In addition, Supplementary section S1 gives further results on a synthetic graph classification dataset designed to test the assumptions underlying our model. We first comment on the common features of the architectures tested (including implementation and compute time), before outlining the specific experimental setup and results for each task.

Datasets: LLPS occurs when multivalent interactions between protein chains cause the formation of a distinct phase of protein condensates. The ability of proteins to form condensates through LLPS is believed to be largely driven by the enrichment of certain binding domains in the molecule, such as the intrinsically disordered regions (IDRs) [8]. IDRs [9] are considered to be segments that lack unique 3D structure. In our protein graphs, each amino acid is represented by a node. The edges of the graph reflect the spatial closeness of the amino acids in the original molecule. We prepare three LLPS dataset focusing on regulator and scaffold proteins. The first dataset is for regulators with ground truth PDB 3D structure. For positive examples in this regulator dataset, we use the 135 proteins labeled as regulators in the DrLLPS dataset [36]. We kept only those proteins for which the spatial coordinate information of at least 70% of the chain is present in the respective PDB file. If multiple PDB files are linked to one protein, we choose the one with the highest spatial resolution. Our dataset contains an equal number of negative examples, which are selected randomly from Uniprot [37] such that they are not found in the DrLLPS dataset, and their PDB files are required to contain the full chain spatial information. The second dataset is for regulators with protein graphs constructed from AlphaFold predicted 3D structure. This dataset contains nearly identical sets of regulator proteins as the previous one, while the latter uses the ground truth PDB 3D structure to construct the protein graphs. For the scaffold dataset, we use 137 proteins labeled as scaffold in the DrLLPS dataset [36] with existing AlphaFold predicted 3D structure information on UniProt webpage. We then select an equal number of negative samples that are not found in the DrLLPS dataset and have AlphaFold predicted 3D structure information on UniProt webpage. See supplementary material for full list of proteins used. We divide the data into 40% training, 30% validation and 30% test set.

Model Architectures and training: Three classes of models are considered in this work: a basic GNN (BG), Clustering+GNN (CG) and *Pre-Linear*+Clustering+GNN (PCG). We further divide these classes into subcategories based on the message passing operator used, including GCNConv [33] and TransformerConv [34] based models, and name them based on the combination of the class and the message passing operator. For GCNConv-based models, the subcategories were named BGCN, CGCN, and PCGCN. For TransformerConv-based models, the subcategories were named BGTN, CGTN, and PCGTN.

The BG models and the GNN module embedded in CG and PCG models have four layers. They use the Adam optimizer with a learning rate of 0.008, a batch size of 30, and the dimension of the node features of all hidden layers is set to 25. The maximal number of epochs is set to 300 and the validation accuracy is monitored with a patience parameter of 160 to determine early stopping. This set of hyperparameters was found to be optimal for the stand alone basic GCN model on LLPS dataset. For the CG and PCG models, we use AP with a pre-defined similarity matrix to perform the clustering. The off-diagonals of the similarity matrix are calculated with eq. (3). The diagonals are filled with S_m, the median of all S_{ij}, for $i \neq j$. The damping factor of the AP is set to 0.88 and the maximum number of iterators is set to 1600. We set the preference parameter of the AP to $p_{\text{fac}} \times S_m$. We test various combinations for $cp = (p_{\text{fac}}, C_1, C_2)$. Recall that C_1 and C_2 are factors from eq. (3) controlling the relative importance of different terms. We noticed that increasing p_{fac} will result in larger average cluster sizes and increasing C_1 leads to higher node degree.

For all PCG models, SBO is used for the PLL training as described in sec. 2.2. In each meta-epoch, the number of random samples is set to $N_S = 15$. The score function used to update μ and σ is derived from the CrossEntropyLoss of the training dataset $L_{\text{CE}}(\text{train})$ according to $\max(-L_{\text{CE}}(\text{train}) + 0.6, 0)$. In our implementation, each meta-epoch consists of an update of μ followed by an update of σ.

For basic GCN/GTN and random dropout GCN/GTN, we report the average accuracy over all samples as the performance. For CG and PCG models, we report the test accuracy of the model with the best validation accuracy across all meta-epochs, as the top model performance. The average test accuracy of the meta-epoch with the highest average validation accuracy is reported as the average performance. To allow an exact comparison with the PCG models, we separate the results of the CG models also into groups, each consisting of 15 samples.

The node features are chosen according to the nature of LLPS based on experimental evidence and existing physical models (see [38, 39]). We use five features including charge [40], hydrophobicity [41], polarity [42], flexibility index [43] and IDR scale [44] to represent amino acids in the molecular graph. We repeat the training of BG models with 105 random initial parameters. For the CG models, the training is repeated 105 times and separated into 7 groups in analogy to the meta-epochs. For PCG models, we train the models for 60 meta-epochs. We initialize μ such that the PLL is equal to identity (equivalent to CG) and $\sigma_{t=0} = \sqrt{3}$. At the end of the training, we observe that the final σ around $t = 60$ is typically well below 0.1. During our hyper-parameter grid search for *cp*-parameters, we covered a range of cluster sizes from less than 10 to over 60 nodes and approximately 10% to 63% of the full graph node degree for regulator proteins. For the scaffold proteins, we tested a much larger range of *cp*-parameters and covered cluster sizes from ≈ 10 to ≈ 70 and $\approx 20\%$ to $\approx 80\%$ of the original graph node degree.

Implementation and compute time: Open-source python libraries including Torch, PyTorch-geometric and NetworkX are used in the implementation. The message passing operators TransformerConv and GCNConv from torch_geometric.nn are used. The AP is performed using the function sklearn.cluster.AffinityPropagation. The protein graph is generated with the python package *proteingraph*[45]. With respect to compute time, one meta-epoch for one sample takes ~ 3 hours on a CPU with 32G. For the PCG model, 60 meta-epochs for one *cp* with $N_S = 15$ required approximately 2500 CPU-hours. For the CG models, ~ 400 CPU-hours were required for one *cp* setting.

Table 1: Regulators with ground truth PDB structure information. The test accuracy is shown for both basic GCN and GTN, as well as for random dropout GCN and GTN.

drop %	basic (0%)	50%	70%	90%
GCN	0.65 ± 0.042	0.652 ± 0.044	0.656 ± 0.045	0.675 ± 0.059
GTN	0.654 ± 0.047	0.65 ± 0.044	0.632 ± 0.046	0.645 ± 0.055

Table 2: Regulators with ground truth PDB structure information. Average performance of top 3 *cp*-parameters for PCGCN and PCGTN is presented. 'meta/raw' node degree (cluster size) denotes the average node degree (cluster size) obtained by using the optimized meta-features or raw features as input for the clustering algorithm.

	val. accu.	test accu.	meta/raw node degree	meta/raw cluster size	CG test accu
			PCGCN		
2, 0.1, 0.2	0.821 ± 0.036	0.681 ± 0.042	0.551/0.615	30.1/ 23	0.652 ± 0.052
2, 0.1, 0.1	0.817 ± 0.021	0.716 ± 0.036	0.67/0.667	42.5/24.9	0.663 ± 0.049
1, 0.2, 0.1	0.813 ± 0.023	0.7 ± 0.041	0.593/1.101	19.2 / 8.19	0.681 ± 0.06
			PCGTN		
1, 0.1, 0.1	0.824 ± 0.028	0.668 ± 0.033	0.466/0.572	27.36/13.13	
1, 0.2, 0.1	0.815 ± 0.022	0.67 ± 0.024	0.524/1.101	19.2 / 8.19	
1, 0.1, 0.4	0.79 ± 0.036	0.672 ± 0.051	0.362/0.382	20.3/15	

3.1 Model performance on PDB and AlphaFold regulator datasets

In this section, we apply our model for the classification of regulator proteins. For nearly the same set of regulator proteins, we first use as input protein graphs constructed with ground truth PDB 3D structure. Following this, we test our model on regulator proteins using AlphaFold predicted 3D structure to construct the protein graphs.

In table 1, we summarize the performance of basic GCN/GTN models and random dropout models as the baseline performance of classification of regulator proteins using graphs constructed with (noisy) ground truth PDB 3D structures. The average validation/test accuracy of the basic GCN is measured to be $0.724 \pm 0.03/0.65 \pm 0.042$ and for the basic GTN it is $0.693 \pm 0.029/0.654 \pm 0.047$.

For this dataset of ground truth PDB 3D structure, the range of $cp = (p_{\text{fac}}, C_1, C_2)$ settings explored is: $p_{\text{fac}} \in \{1, 2, 3, 4, 5, 6\}$, $C_1 \in \{0.1, 0.2, 0.4, 0.8\}$ and $C_2 \in \{0.1, 0.4\}$. In table 2, we show the average performance of the best three choices of *cp*-parameters for this dataset. Since the difference between the validation accuracies of these models is within the error range and the *cp*-values are very close to each other, we report the average test accuracy of the top three models as the final test accuracy. We found that the PCGCN achieved an average validation/test accuracy of 0.817/0.699 and a test AUC of 0.749, and overperformed the 0.81/0.67 validation/test accuracy of the PCGTN. For the best *cp*-parameters for the PCGCN, we also tested the CGCN models, namely using raw features as input for clustering instead of meta-features. The results are also listed in table 2, the average test accuracy of CGCN models is 0.665, which is higher than the basic GCN models but lower than PCGCN models where a linear layer is implemented before the clustering. Thus, on the regulator protein dataset with ground truth PDB structures, our adaptive cluster-based architecture outperforms models using the original graph (i.e. the basic GCN/GTN models), random dropout GCN/GTN, and models which use a fixed sparsification strategy (i.e. the CGCN, where W^{meta} is not learned, and thus no meta-features are used).

The performance of our model on the regulator dataset but using AlphaFold predicted 3D structure is presented in table 3. We performed a grid search for *cp*-parameters with $p_{\text{fac}} \in [7, 9]$, $C_1 \in [2, 4]$ and $C_2 = 0.1$, and we also tested the nine most promising *cp*-values from the scaffold AlphaFold dataset (see below section 3.2). Similar to our calculation of the performance of our model on regulator proteins with ground truth 3D structure information, we average over the three models with the highest average validation accuracies. One should

Table 3: Regulators with AlphaFold predicted 3D structure. Average performance of the three *cp*-parameters with the highest validation accuracy for PCGCN and PCGTN is presented.

PCGCN			**PCGTN**		
cp	val. accu.	test accu.	*cp*	val. accu.	test accu.
8, 2.5, 0.1	0.722 ± 0.027	0.675 ± 0.045	7, 3.0, 0.1	0.737 ± 0.028	0.693 ± 0.032
7, 3.0, 0.1	0.718 ± 0.018	0.677 ± 0.042	8, 2.5, 0.2	0.736 ± 0.028	0.695 ± 0.044
6, 2.5, 0.1	0.714 ± 0.029	0.674 ± 0.03	8, 3.5, 0.1	0.734 ± 0.044	0.726 ± 0.022
basic GCN	0.686 ± 0.038	0.656 ± 0.036	basic GTN	0.698 ± 0.03	0.671 ± 0.03

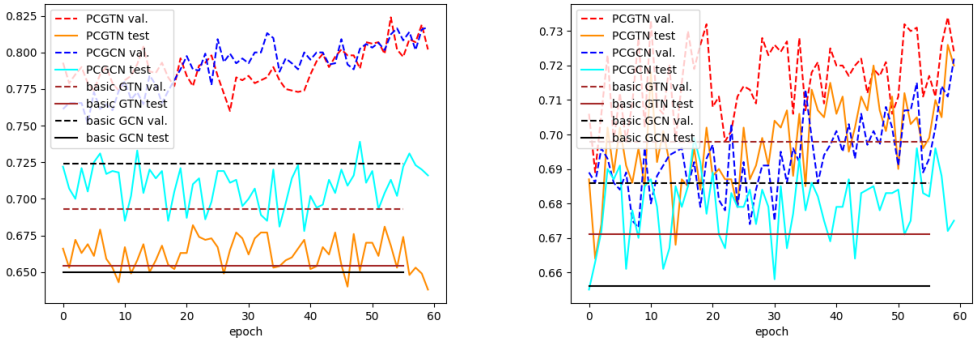

Figure 2: For basic GCN and GTN models, we show a simple average over all samples. Left: Regulator proteins with ground truth PDB 3D structure. The averaged validation (dashed) and test (solid) accuracy are shown for PCGCN *cp*=(2,0.1,0.1) and PCGTN $cp = (1, 0.1, 0.1)$. The best meta-epoch of PCGTN is 53 and the best meta-epoch of PCGCN is 59. Right: Regulator proteins with AlphaFold predicted 3D structure. For PCGCN, $cp = (8, 2.5, 0.1)$ is shown with the best epoch at 59. For PCGTN, $cp = (8, 3.5, 0.1)$, the best epoch is found at epoch 58.

note that the best *cp*-parameters are among the most promising ones on the scaffold dataset. For the PCGTN, which performed better than the PCGCN on this dataset, our model on regulator proteins using AlphaFold predicted 3D structure achieved a validation/test accuracy of 0.736/0.705 and a test AUC of 0.786, better than the performance on ground truth PDB 3D structures.

These results may reflect the interplay between the quality / noise characteristics of the data, and the complexity of the model. For instance, the AlphaFold predicted structures are less noisy than the PDB structures (although potentially with biases introduced by the AlphaFold method), which may plausibly allow the PCGTN models to achieve better performance since they are less prone to over-fitting in this context than the noisy PDB dataset. The latter, however, may favor the PCGCN, which contains fewer parameters.

In addition, we show the average validation and test accuracy as a function of meta-epochs t for one PCGCN and one PCGTN model for the two datasets using different protein graphs in fig. 2, where the basic GCN and basic GTN performances are also shown as baselines. The validation accuracies of our adaptive models shows a clear upwards trend especially towards the later stages of the training, suggesting that the PLL, and thus meta-features, are successfully trained through our variational optimization. The gaps between the validation accuracies and test accuracies in the left panel of fig. 2 indicate overfitting. This is not observed on the dataset using AlphaFold predicted 3D structure information. As discussed above, this may reflect the advantages of the more flexible PCGTN architecture on the less noisy AlphaFold structures.

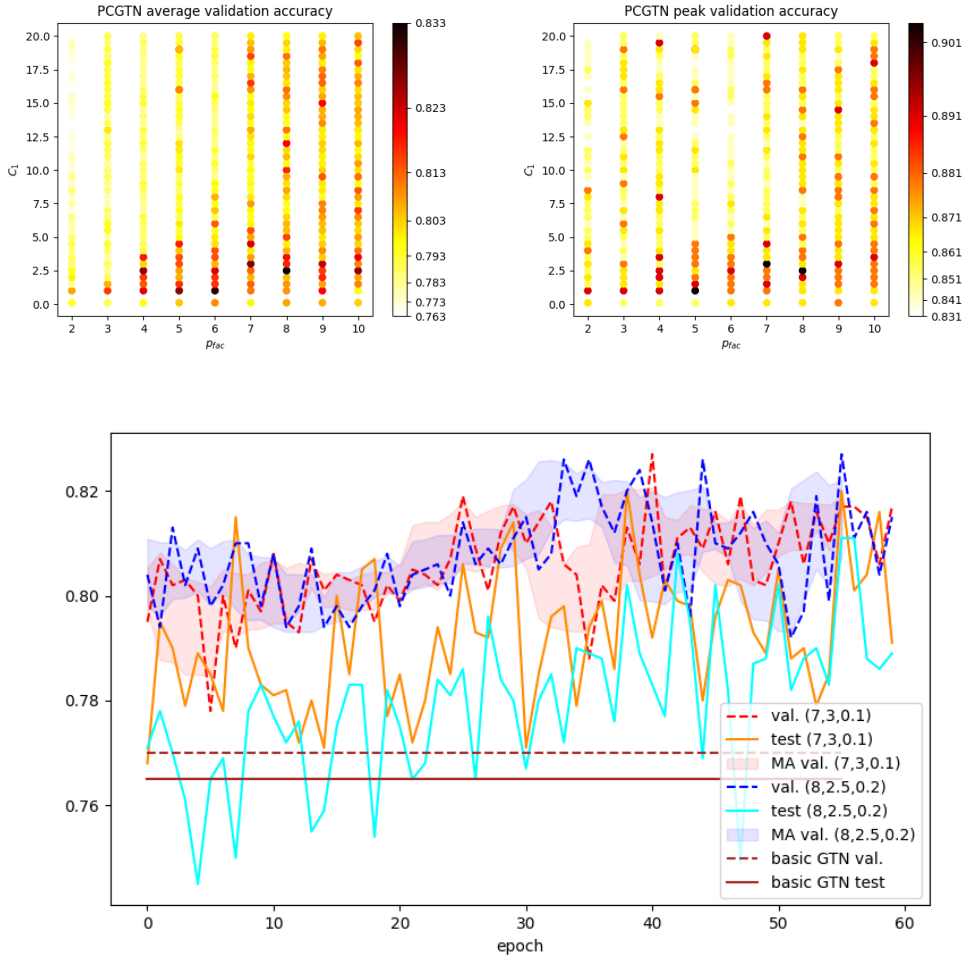

Figure 3: 2D plot showing p_{fac} and C_1 dependency of the validation accuracy of the PCGTN at $C_2 = 0.1$; the optimal C_2 value is only screened using the most promising p_{fac} and C_1 choices. Left: average validation accuracy in the best performing epoch. Right: Peak validation accuracy. Bottom: Average validation accuracy and test accuracy against number of meta-epochs for the best model $cp = (8, 2.5, 0.2)$ and the second best model $cp = (7, 3.0, 0.1)$ compared with the performance of the basic GTN and PCGTN models, show clear improvement of the PCGTN through the variational optimization and gradual outperformance with respect to the basic GTN. The best meta-epoch of $cp = (8, 2.5, 0.2)$ in this example is found to be 55 and for $cp = (7, 3.0, 0.1)$ it is meta-epoch 40.

3.2 Model performance on AlphaFold scaffold dataset

In this section we present the performance of our model on scaffold proteins using AlphaFold predicted 3D structure. A similar test on scaffold proteins using ground truth PDB 3D structure was not possible due to the very limited number of known scaffold proteins for which the ground truth 3D structure is available. To find the optimal *cp*-parameters $(p_{\text{fac}}, C_1, C_2)$, we first performed a comprehensive hyper-parameter screening for $p_{\text{fac}} \in [2, 10]$, $C_1 \in [0.1, 20]$ while keeping $C_2 = 0.1$. Then, for several p_{fac} and C_1 choices with the highest validation accuracies, we screen for C_2 over a range of $C_2 \in [0.1, 3]$.

We observed the average validation and test accuracy of all (105) basic GCN to be $0.779 \pm 0.027/0.731 \pm 0.052$. For the basic GTN, this is observed to be $0.77 \pm 0.029/0.766 \pm 0.045$.

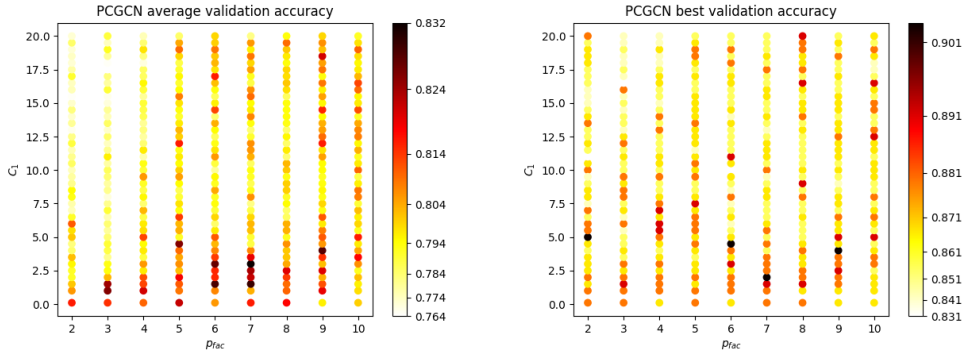

Figure 4: 2D plot showing p_{fac} and C_1 dependency of the validation accuracy of the PCGCN at $C_2 = 0.1$; the optimal C_2 value is only screened using the best p_{fac} and C_1 choices. Left: average validation accuracy in the best performing epoch. Right: Peak validation accuracy.

In fig. 3, the validation accuracies resulting from our parameter screening for PCGTN p_{fac} and C_1 is shown. The best region for both the average performance and peak performance is identified to be $p_{\text{fac}} \in [5, 8]$ and $C_1 \in [1.0, 3.0]$. For the top performing p_{fac} and C_1 values in this region, we continued to perform a screening over $C_2 \in [0.1, 3]$. Among the top performing models, we found a small region densely filled with many of the best validation accuracies; this region included $cp = (9, 3, 0.1), (8, 2.5, 0.1), (8, 2.5, 0.2), (8, 2.5, 0.3), (8, 2.5, 0.4), (8, 3, 0.1), (8, 3.5, 0.1), (7.0, 3.0, 0.1)$ and $(6.0, 2.5, 0.1)$. Since the difference between the validation accuracies among the top 20 models is less than 2%, which is less than the error range, we identify the small region centered around $cp = (8, 2.5, 0.2)$ to be the best cp-parameter choice. For $cp = (8, 2.5, 0.2)$, our model achieved an average training accuracy of 0.902 ± 0.083, an average validation accuracy of 0.827 ± 0.021, an average test accuracy of 0.811 ± 0.035 and an average test AUC of 0.894 ± 0.033 in the best performing meta-epoch. This outperforms the test accuracy of the basic GTN 0.765 ± 0.045, random dropout GTN 0.752 ± 0.058 (34%, similar node degree as $(8, 2.5, 0.2)$) and the result using raw features of the proteins (CGTN) with validation accuracy 0.7831 ± 0.0271 and test accuracy 0.696 ± 0.057. We also found the peak performing model to be $cp = (8, 2.5, 0.2)$ with a validation accuracy of 0.892, a test accuracy of 0.841 and a test AUC of 0.92.

In addition to the PCGTN model, we provide here also the results of the PCGCN models. As shown in fig. 4. The best cp-parameters for the PCGCN are found to be in the region $p_{\text{fac}} \in [5, 7]$ and $C_1 \in [0, 5]$. Through additional C_2 screening in this region, we found the best cp-parameter to be $(7, 3.0, 0.1)$ with a test accuracy of 0.746.

We observed that the average performance of the top 20 models for PCGTN is slightly higher that of PCGCN. We also note that for the AlphaFold predicted structures, the more flexible transformer-based model (see in fig. 3) no longer has the over-fitting problem caused by the noisy nature of the ground truth PDB graph as in section 3.1. Combining these considerations, we conclude that PCGTN achieves the best overall performance in the region around $cp = (8.0, 2.5, 0.2)$, agreeing with the results on the AlphaFold regulator dataset above, which showed the PCGTN to perform better than PCGCN when using AlphaFold as opposed to PDB structures, since the PCGTN, with its increased model complexity, can exploit the less noisy datasets generated by AlphaFold.

3.3 Comparison to previous predictors

Previous methods for phase separation protein (PSP) prediction [46, 47, 48, 49] have concentrated on predicting scaffold proteins. To our knowledge, the optimal performance using previous methods is a recall of 10.54% on regulators and 46% on scaffolds, reported in Ref. [48] at an acceptance rate of 0.02 for the background class (here, the whole proteome). We thus applied our optimal models and ranked each protein in our test set by the final score ($W^{\text{final}} \sum_i \mathbf{x}_i^L$). We taken into account the results of all 15 samples in the epoch

of the best average validation accuracy, and found the recall at a 0.02 acceptance rate to be $23.6 \pm 6.8\%$ for the PCGTN model with $cp = (8, 3.5, 0.1)$ trained with regulators with AlphaFold predicted 3D structure and $50.6 \pm 16.4\%$ for scaffolds using PCGTN with $cp = (8, 2.5, 0.2)$, suggesting that our model achieves better performance than prior methods for the prediction of regulators and scaffolds. See Supplementary section S2.2 for further details.

3.4 Properties of the learned clusters

Since we observed above that the clustered model performed best overall, we were interested in the overall properties of the subgraphs discovered by these models.

The three best PCGCN models on regulator proteins with ground truth 3D structure with optimized PLL have a singleton percentage of 65.12%, 59.92% and 63.72%. Although the average overall node degree is only around 0.6, the average node degree per cluster after singleton removal is $2.13, 2.45$ and 2.13. For the optimized PCGTN model $cp = (8, 3.5, 0.1)$ on regulator proteins with AlphaFold 3D structure the singleton percentage is 70.4%. The average node degree per cluster after singleton removal is 1.82. The singletons, defined as nodes without edges, can be interpreted as potentially noisy nodes, thus the model may have learned to prohibit the information propagation to their neighbors. It is worth mentioning that the optimal *cp*-combinations of the AlphaFold dataset is different from the ones using PDB ground truth structure information. This is not surprising since the AlphaFold predicted structures only include the chain of interest, while PDB graphs often include unwanted information from other chains in the complex, or multiple conformations of the same chain.

We noticed that the cluster statistics of the best models for the regulator proteins using AlphaFold predicted 3D structure are very similar to that of regulators with ground truth PDB structure information despite the distinct *cp*-parameters. We believe that the stability of the cluster statistics in these tests using quite different graphs reflects underlying aspects of the LLPS regulator protein biology. We note that this range of node degree is consistent with the nature of the lack of stable structure in IDRs, which is a result of missing bonds that are responsible for maintaining the 3D structure of proteins in shape. In case of a linear chain with only backbones, the average node degree is 2, which is close to the values observed in the best models above.

For the scaffold protein dataset using AlphaFold predicted structure, the best performing model (PCGTN with $cp = (8.0, 2.5, 0.2)$) measured an average node degree of 2.1 and an average node degree per cluster after singleton removal of 4.8 (the singleton percentage is 24.76%).

We noticed that the best models, despite having different *cp*-values, underlying structure information and message-passing schemes, demonstrate similar cluster statistics on refined graphs through the adaption of the PLL parameters (see table 2, which compares raw node degree and raw cluster size to meta node degree and meta cluster size). Further, we note that our comprehensive grid search of *cp*-parameters in the scaffold results shows a gradual change in performance, and reveals clearly the presence of a stable optimal region of the clustering parameters leading to enhanced classification accuracies, suggesting the above statistics are stable for the optimal models.

4 Validation against annotated protein regions

In this section, we assess the biological relevance of the clusters identified by our model, by comparing them against annotations from UniProt, a comprehensive resources for protein sequence and annotation data. Specifically, we analyze the performance of the PCGTN model on the scaffold dataset, demonstrating its proficiency in identifying biological significant features crucial to protein functionality. UniProt provides several vital annotations, notably the identification of disordered protein regions. These regions, lacking a stable three-dimensional structure under physiological conditions, play significant roles in protein function despite their unordered nature.

For each protein, the model identifies n clusters, while UniProt provides m disordered regions. The overlap score and cluster size error are calculated as follows (refer to Supplementary section S3.2 for alternative methods):

Overlap assessment using Jaccard Index

- **Procedure**: For each disordered region from UniProt, we calculate the Jaccard Index with every cluster identified by the PCGTN model. The Jaccard Index, a measure of set similarity, is computed as the size of the intersection divided by the size of the union of the disordered region and the cluster.
- **Selection of maximum indices**: The highest Jaccard Index is selected for each disordered region, representing the best match with the model's clusters.
- **Weighted average calculation**: We compile m maximum Jaccard Indices, each corresponding to a disordered region from UniProt. The final overlap score is the weighted average of these indices. Weights are proportional to the length of the disordered regions, giving greater importance to larger regions.

Cluster size ratio error analysis

- **Error calculation for size compatibility:** We assess the size compatibility between the model's clusters and the disordered regions by calculating the cluster size ratio error. This error is the absolute value of the difference between 1 and the cluster size ratio, defined as the size of the cluster divided by the size of the disordered region.
- **Selection and weighting of errors:** The smallest error for each disordered region is selected. The final cluster size ratio error score is the weighted average of these smallest errors, with weights based on the length of the disordered regions.

We compared the overlap score distributions between proteins in the signal (LLPS) group (labeled as scaffold in the DrLLPS dataset) and the background group (not present in the DrLLPS dataset). We divided the score range [0,1] into 50 bins, analyzing the relative frequency of scores within each bin.

Our biological interpretation considered a subset of parameters, $p_{\text{fac}} \in [2, 10]$ and $C_1 \in [0.1, 10]$ (half the range of the parameter screening we did for the performance test) while keeping $C_2 = 0.1$, then an additional screening for C_2- in the range $C_2 \in [0.1, 3]$ for the top-performing parameters as described in section 3.2.

Various metrics for different p_{fac} and C_1, and $C_2 = 0.1$ are shown in fig. 5: the median difference between in overlap score between signal and background proteins in fig. 5(a), right-hand Mann-Whitney p-value for the hypothesis that signal proteins have higher overlap scores than background proteins in fig. 5(b), and the mean difference in the distribution of the absolute value of 1 minus the cluster size ratio, indicating optimal size match fig. 5(c). Examples of overlap score distributions for three $cp-$parameters with the best performance is shown in fig. 5(d). We note that for $C_2 \neq 0.1$, significant p-values and positive median differences between signal and background proteins are observed from the overlap score of the cp-parameters that correspond to the best performing models. Refer to Supplementary section S3.1 for further results regarding $C_2 \neq 0.1$.

In fig. 5 (a), for the majority of the *cp*-parameters, the median overlap score of signal proteins is larger than that of the background proteins. In (b), the p-value is also significant for almost the entire parameter space. In (c), we observed that for the majority of $cp-$parameters, the sizes of detected clusters is closer to the size of disordered regions for signal proteins than background proteins. In addition, we observe a gradual increase of the difference between the signal and background proteins in (a), a gradual increase of the significance of the hypothesis that the overlap score distribution of signal protein is larger than the background proteins in (b), and an increased difference between how close the cluster sizes and disordered region sizes are in the signal and background proteins in (c) towards the lower right triangular part of the parameter space. This pattern of a higher correspondence between detected clusters and annotated disordered regions, as well as the region densely filled with significant p-values ($p_{\text{fac}} = [6, 8]$, $C_1 \in [2, 5]$, and the region corresponding to the best-performing models around

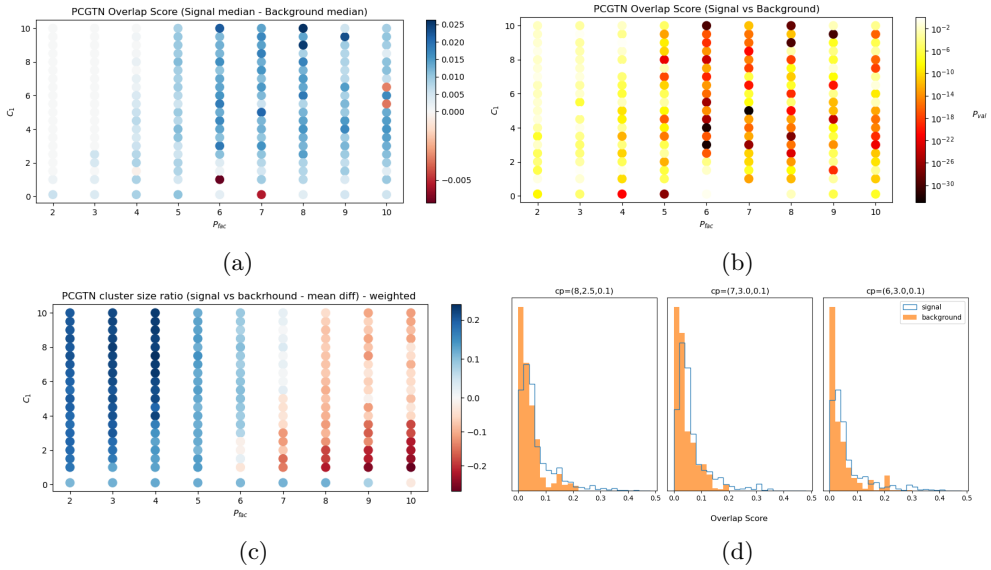

Figure 5: (a) The difference between median overlap score from signal (LLPS) and background proteins. Zeros are in white indicating no difference between medians of two group, positive values are in blue indicating signal group has a larger median, negative values are in red indicating background group has a larger median. (b) Overlap score right hand Mann-Whitney p-value, darker color indicating a smaller p-value as more significant that signal group has a larger overlap score than the background group. (c) The difference between mean value of | 1- cluster size ratio | of signal and background proteins. Zeros are in white indicating no difference between two group, positive values are in blue indicating background proteins have cluster size ratio closer to 1, and negative values are in red indicating signal proteins have cluster size ratio closer to 1. (d) For three of the best models from the performance test, we show the distribution of overlap scores for signal and background proteins.

$cp = (8.0, 2.5, 0.2)$ as mentioned in section 3.2) matches the pattern we observed in the performance of our model (see fig. 3).

5 Discussion

In this work, we proposed a formalism to enhance the predictive power of conventional graph neural network by adaptive graph refinement. To this end, we formulated a similarity measure that captures the probability of nodes pairs jointly exhibiting characteristic patterns within larger systems. We used this measure to remove the connections between nodes that are unlikely to be related for the given task. To enhance the flexibility of our model in adapting to diverse tasks, we leveraged a linear layer to extract meta-features. The usage of meta-features enables the training of the similarity measure with the task labels, thereby facilitating task-specific adjustments in the model's predictions. We have shown on multiple datasets that our model achieves state-of-the-art performance for protein LLPS prediction, including regulator and scaffold LLPS proteins, with PDB and AlphaFold generated graph-structures.

For all the datasets considered in this work, adding a clustering layer based on the similarity measure we formulated and a PLL improves the test accuracy by at least 5% compared to conventional graph neural network models. From the model architecture point of view, further investigation of alternative clustering approaches and similarity measures (incorporating, for instance, node interaction or amino acid k-mer terms) may lead to further gains in performance. From the practical training side, increasing the number of samples in the variational optimization, increasing the number of meta-epochs, and additional optimization of network architecture and hyperparameter settings could further improve the performance of the model.

To our best knowledge, our model is the first graph-based predictor for LLPS, demonstrating superior performance compared to existing LLPS predictors in the task of predicting both regulator and scaffold proteins. In addition, we interpreted and validated the clusters identified by our model by aligning them with biological annotated protein disordered regions. Notably, the detected clusters successfully match these important biological regions, which may be the principal reason for the improved prediction accuracy. The capability of our model to identify meaningful underlying subgraphs can be further employed to gain additional insights into systems of interest. As a next step, we plan to explore novel protein regions identified by our model, which are not aligned with annotated protein regions already known to be associated with LLPS proteins, in order to further advance our understanding of this biological phenomenon.

Code availability

The source code to run the variational graph partitioning GNN is available at `https://github.com/gaoyuanwang1976/GraphPartition_SBGNN`.

Acknowledgement

We acknowledge support from the NIH and from the AL Williams Professorship funds. We would also like to thank Lawrence H. Phillips for valuable discussions.

References

[1] Anthony A. Hyman, Christoph A. Weber, and Frank Jülicher. "Liquid-Liquid Phase Separation in Biology". In: *Annual Review of Cell and Developmental Biology* 30.1 (2014), pp. 39–58.

[2] Bin Wang et al. "Liquid–liquid phase separation in human health and diseases". In: *Signal Transduction and Targeted Therapy* 6.1 (Aug. 2021).

[3] W. Michael Babinchak and Witold K. Surewicz. "Liquid–Liquid Phase Separation and Its Mechanistic Role in Pathological Protein Aggregation". In: *Journal of Molecular Biology* 432.7 (2020), pp. 1910–1925. ISSN: 0022-2836.

[4] Steven Boeynaems et al. "Protein Phase Separation: A New Phase in Cell Biology". In: *Trends in Cell Biology* 28.6 (June 2018), pp. 420–435.

[5] Pei-Hua Peng, Kai-Wen Hsu, and Kou-Juey Wu. "Liquid-liquid phase separation (LLPS) in cellular physiology and tumor biology". en. In: *American Journal of Cancer Research* 11.8 (Aug. 2021), pp. 3766–3776.

[6] Xuhui Tong et al. "Liquid–liquid phase separation in tumor biology". In: *Signal Transduction and Targeted Therapy* 7.1 (July 2022).

[7] Solomiia Boyko and Witold K. Surewicz. "Tau liquid–liquid phase separation in neurodegenerative diseases". In: *Trends in Cell Biology* 32.7 (July 2022), pp. 611–623.

[8] Wade Borcherds et al. "How do intrinsically disordered protein regions encode a driving force for liquid–liquid phase separation?" In: *Current Opinion in Structural Biology* 67 (Apr. 2021), pp. 41–50.

[9] M Madan Babu. "The contribution of intrinsically disordered regions to protein function, cellular complexity, and human disease". In: *Biochemical Society Transactions* 44.5 (Oct. 2016), pp. 1185–1200.

[10] Robert McCoy Vernon et al. "Pi-Pi contacts are an overlooked protein feature relevant to phase separation". In: *eLife* 7 (Feb. 2018). Ed. by Yibing Shan, e31486. ISSN: 2050-084X.

[11] V N Uversky, J R Gillespie, and A L Fink. "Why are "natively unfolded" proteins unstructured under physiologic conditions?" In: *Proteins* 41.3 (Nov. 2000), pp. 415–427.

[12] Hermann Broder Schmidt and Dirk Görlich. "Nup98 FG domains from diverse species spontaneously phase-separate into particles with nuclear pore-like permselectivity". In: *eLife* 4 (Jan. 2015). Ed. by Karsten Weis, e04251. ISSN: 2050-084X.

[13] Wenqi Fan et al. "Graph Neural Networks for Social Recommendation". In: *The World Wide Web Conference*. ACM, May 2019.

[14] Weiwei Jiang and Jiayun Luo. "Graph neural network for traffic forecasting: A survey". In: *Expert Systems with Applications* 207 (Nov. 2022), p. 117921.

[15] Xiao-Meng Zhang et al. "Graph Neural Networks and Their Current Applications in Bioinformatics". In: *Frontiers in Genetics* 12 (July 2021).

[16] Michelle M. Li, Kexin Huang, and Marinka Zitnik. "Graph representation learning in biomedicine and healthcare". In: *Nature Biomedical Engineering* 6.12 (Oct. 2022), pp. 1353–1369.

[17] Yuyang Wang, Zijie Li, and Amir Barati Farimani. "Graph Neural Networks for Molecules". In: *arXiv preprint arXiv:2209.05582* (2022).

[18] Alex Fout et al. "Protein Interface Prediction using Graph Convolutional Networks". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[19] Vladimir Gligorijević et al. "Structure-based protein function prediction using graph convolutional networks". In: *Nature Communications* 12.1 (May 2021).

[20] Zuobai Zhang et al. "Protein Representation Learning by Geometric Structure Pretraining". In: *The Eleventh International Conference on Learning Representations*. 2023.

[21] Yu Rong et al. "DropEdge: Towards Deep Graph Convolutional Networks on Node Classification". In: *International Conference on Learning Representations*. 2020.

[22] Yu Chen and Lingfei Wu. "Graph neural networks: Graph structure learning". In: *Graph Neural Networks: Foundations, Frontiers, and Applications* (2022), pp. 297–321.

[23] Darwin Saire Pilco and Adın Ramırez Rivera. "Graph learning network: A structure learning algorithm". In: *ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data*. 2019.

[24] Yu Chen, Lingfei Wu, and Mohammed Zaki. "Iterative deep graph learning for graph neural networks: Better and robust node embeddings". In: *Advances in neural information processing systems* 33 (2020), pp. 19314–19326.

[25] Cheng Zheng et al. "Robust graph representation learning via neural sparsification". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 11458–11468.

[26] Petar Veličković et al. "Graph Attention Networks". In: *International Conference on Learning Representations*. 2018.

[27] Will Grathwohl et al. "Backpropagation through the Void: Optimizing control variates for black-box gradient estimation". In: *International Conference on Learning Representations*. 2018.

[28] Mohammad Khan et al. "Fast and scalable bayesian deep learning by weight-perturbation in adam". In: *International conference on machine learning*. PMLR. 2018, pp. 2611–2620.

[29] Hossein Mobahi and John Fisher III. "A theoretical analysis of optimization by Gaussian continuation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 29. 1. 2015.

[30] Marius Leordeanu and Martial Hebert. "Smoothing-based optimization". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pp. 1–8.

[31] Jairo Rojas-Delgado et al. "Hyper-parameter Optimization Using Continuation Algorithms". In: *Metaheuristics*. Springer International Publishing. 2023, pp. 365–377. ISBN: 978-3-031-26504-4.

[32] Arjan Kuijper, Luc MJ Florack, and Max A Viergever. "Scale space hierarchy". In: *Journal of Mathematical Imaging and Vision* 18 (2003), pp. 169–189.

[33] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations*. 2017.

[34] Yunsheng Shi et al. "Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification". In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. Ed. by Zhi-Hua Zhou. Main Track. International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 1548–1554.

[35] Brendan J Frey and Delbert Dueck. "Clustering by passing messages between data points". In: *science* 315.5814 (2007), pp. 972–976.

[36] Wanshan Ning et al. "DrLLPS: a data resource of liquid–liquid phase separation in eukaryotes". In: *Nucleic Acids Research* 48.D1 (Nov. 2019), pp. D288–D295.

[37] "UniProt: the Universal Protein knowledgebase in 2023". In: *Nucleic Acids Research* 51.D1 (2023), pp. D523–D531.

[38] Rita Pancsa, Wim Vranken, and Bálint Mészáros. "Computational resources for identifying and describing proteins driving liquid–liquid phase separation". In: *Briefings in Bioinformatics* 22.5 (Feb. 2021).

[39] Gregory L Dignon, Wenwei Zheng, and Jeetain Mittal. "Simulation methods for liquid–liquid phase separation of disordered proteins". In: *Current Opinion in Chemical Engineering* 23 (Mar. 2019), pp. 92–98.

[40] JEAN-LUC FAUCHÈRE et al. "Amino acid side chain parameters for correlation studies in biology and pharmacology". In: *International Journal of Peptide and Protein Research* 32.4 (Jan. 2009), pp. 269–278.

[41] Jack Kyte and Russell F. Doolittle. "A simple method for displaying the hydropathic character of a protein". In: *Journal of Molecular Biology* 157.1 (May 1982), pp. 105–132.

[42] R. Grantham. "Amino Acid Difference Formula to Help Explain Protein Evolution". In: *Science* 185.4154 (Sept. 1974), pp. 862–864.

[43] R. BHASKARAN and P.K. PONNUSWAMY. "Positional flexibilities of amino acid residues in globular proteins". In: *International Journal of Peptide and Protein Research* 32.4 (Jan. 2009), pp. 241–255.

[44] Andrew Campen et al. "TOP-IDP-Scale: A New Amino Acid Scale Measuring Propensity for Intrinsic Disorder". In: *Protein & Peptide Letters* 15.9 (Sept. 2008), pp. 956–963.

[45] `https://github.com/ericmjl/protein-interaction-network/`. Version 0.3.0.

[46] Robert McCoy Vernon et al. "Pi-Pi contacts are an overlooked protein feature relevant to phase separation". In: *eLife* 7 (Feb. 2018).

[47] Gabriele Orlando et al. "Computational identification of prion-like RNA-binding proteins that form liquid phase-separated condensates". In: *Bioinformatics* 35.22 (Apr. 2019). Ed. by John Hancock, pp. 4617–4623.

[48] Tanlin Sun et al. "Prediction of liquid-liquid phase separation proteins using machine learning". In: *BioRxiv* (2019), p. 842336.

[49] Xiaoquan Chu et al. "Prediction of liquid–liquid phase separating proteins using machine learning". In: *BMC Bioinformatics* 23.1 (Feb. 2022).