MP-GNN: Graph Neural Networks to Identify Moonlighting Proteins

ABSTRACT

Moonlighting proteins are those proteins that perform more than one distinct function in the body. They are pivotal in various metabolic pathways and disease mechanisms. Identifying moonlighting proteins remains a challenge in Computational Biology. In this work, we propose the first graph neural network based models to identify moonlighting proteins. Our models work on large protein-protein interaction (PPI) networks with sparse labels of moonlighting and non-moonlighting proteins. In addition to PPI network, the models make use of features derived from the amino acid sequences of the proteins.

We propose two frameworks: one as graph classification based on the local neighborhood of the query protein; and the other node classification based on the entire graph. These GNN-based methods outperform traditional machine learning methods that have previously been used for moonlighting prediction. The global full network-based model, operating on *Homo sapiens* data achieves accuracy of 88.4% and F1 score of 88.8%. The local neighborhood method is more lightweight and can be applied to larger protein sets with multiple species.

CCS CONCEPTS

• **Applied computing** → **Computational proteomics**.

KEYWORDS

Moonlighting Proteins Prediction, Graph Machine Learning, Graph Neural networks

ACM Reference Format:

. 2018. MP-GNN: Graph Neural Networks to Identify Moonlighting Proteins. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In the 1980s, it was discovered that certain crystallin proteins occurring in eye lenses have amino acid sequences that match enzymes expressed in other tissues [29, 30]. Experiments confirmed that they act like the enzymes in metabolic reactions, thus demonstrating that the same proteins expressed from the same genes could perform very different functions depending on where it is expressed [26]. These dual function proteins were later given the name *Moonlighting proteins*. Since then, the canonical functions of moonlighting proteins have been often found to be associated with critical biological pathways such as Krebs cycle or glycolysis. They are also frequently associated with pathogen virulence [3]. Thus, identification and understanding of moonlighting proteins is important for fundamental understanding of biology, of bacterial activity and drug design [10].

Moonlighting proteins (abbreviated MPs) have been discovered in all kinds of living organisms, but the experimental methodologies to find proteins with multiple functions remain challenging and expensive. According to MoonProt [20] – a database for Moonlighting proteins – only about 500 MPs have been identified to date. Thus, even with our knowledge of billions of protein sequences found by current high throughput technologies [22], we need fundamental developments in understanding MPs and in methods for finding them. Computational techniques that identify good candidates to test for moonlighting properties can narrow the field for experimentation and significantly reduce the cost and effort in finding new moonlighting proteins.

Related works. Statistical machine learning methods have been developed for identification of moonlighting proteins. Methods such as MPFit [14] and DextMP [13] have achieved good accuracy by making use of Gene Ontology (GO) and natural language processing based on literature context respectively. However, these features require human analysis and labelling, making them unsuitable for all but a few select proteins. For example, only a small fraction ($\approx 0.1\%$) of proteins in the Uniprot database carry GO annotations. More recent approaches have made use of more universal features – amino acid sequences and Protein-Protein Interaction (PPI) networks. Features derived from amino acid sequences have formed the basis of conventional machine learning models including Deep neural networks (DNNs), Support Vector machine (SVM) and Random Forest (RF) [25]. K-nearest neighbors was investigated in Hu and Du [9] context of MPs. By definition, these models base their predictions on protein structure and ignore interactions. In contrast, PPI based methods base their predictions purely on inter-protein interactions [12, 14]. These methods used statistical measures and bioinformatic techniques, including remote homology searches, co-evolution, and network analysis, to explore the interactomic features from PPI interactions [1, 7, 8, 12, 14]. However, this approach requires a significant amount of domain knowledge and human expertise. Moreover, these networks are not directly compatible with conventional machine learning models.

Our contributions. Graph neural networks have been developed as a class of machine learning models that can learn from graphs as well as vector features (See [36] for a survey). They have been applied to a range of computational biology problems including protein function prediction, structure prediction and PPI link predictions [11, 17, 31–33, 35].

In this paper, we present the first graph neural network (GNN) framework MP-GNN – designed to identify moonlighting proteins. The models are trained on a PPI network, where the nodes are

Unpublished working draft. Not for distribution.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

labeled with sequence-based features. We study the impact of important graph neural network concepts: Graph convolutions [16], Graph attentions [28] and GraphSAGE sampling [6] are compared.

The neural networks are designed for two different cases (Section 3). First, we introduce Global MP-GNN. When the PPI network consists of proteins of a single species, we devise this global version of MP-GNN which makes use of the full PPI network interconnectivity information, and approach the MP prediction as a node classification task. When analyzing larger networks such as ones with proteins from all species, this approach is impractical and we devise a local version of MP-GNN which is trained on individual node neighborhoods and approach the MP prediction as a graph classification task. To overcome the challenge of small number of labelled proteins, we incorporate a semi supervised learning technique that is found to improve accuracy.

Experiments (Section 4) on the *Homo sapiens* data demonstrate that global MP-GNN outperforms previous methods in terms of accuracy, F-1 score, and precision. The Local MP-GNN models surpass previous methods. The GraphSAGE model with a self-attention pooling (SAGPooling) layer achieved the best performance, with an accuracy of 79.5%, an F-1 score of 79.0%, a precision of 80.0%, a recall of 78.0%, and a ROC-AUC of 75.8%. Using semi-supervised learning, the model achieved even better performance, with an accuracy of 88.4%, a recall of 87.2%, an F-1 score of 88%, a precision of 88.8%, and a ROC-AUC of 81.3%.

Investigating the edge weights predicted by the Graph Attention Network (GAT) model on local ego-networks showed that most self-loops are assigned larger weights, suggesting that GAT prefers to preserve the distribution of weights in neighborhood.

In the following, Section 2 summarizes the basics of graph neural networks and its variants. Readers familiar with these topics may wish to skim it and move ahead to the next section.

2 TECHNICAL BACKGROUND: GRAPH NEURAL NETWORKS

Graph neural networks work by simulating message passing and aggreation between nodes. Suppose $\mathbf{h}_i^{(l)}$ is the feature vector of node i at layer l, then the core of GNNs is to implement the following type of function:

$$\mathbf{h}_i^{(l)} = \phi^{(l)}\left(\mathbf{h}_i^{(l-1)}, \left\{\mathbf{h}_j^{(l-1)}, \forall j \in \mathcal{N}(i)\right\}\right) \tag{1}$$

where $\mathcal{N}(i)$ is the set of neighbours of the node i, ϕ is some aggregate and combine function. That is, at every message passing layer, the features of a node is updated to an aggregate of the features of the previous layers. Through one layer, the features at a node becomes an aggregate of its neighborhood. Over multiple message passing layers, it becomes an aggregate of a multi-hop neighborhood. Depending on the application, the GNN also contains fully connected layers, poooling layers etc. to achieve specific objectives like classification.

2.1 Graph Convolutional Network (GCN)

Graph convolutional networks [16] are a variant of GNNs with a class of layer functions given by:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(W^{(l+1)}\left(\mathbf{h}_i^{(l)} + \sum_{j\in\mathcal{N}(i)} \mathbf{h}_j^{(l)}\right)\right) \tag{2}$$

where W is a learnable weight matrix, and σ is a non-linear activation function such as ReLU. The notable feature here is that the aggregation is linear over the nighborhood, but the fetaure weights are modified by W, followed by the non-linear activation σ.

2.2 Graph Attention Network (GAT)

Unlike GCNs, GATs assign different importances to different edges (or neighbors) [28]. The layers perform the following function:

$$\mathbf{h}_i^{(l)} = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\mathbf{W}\mathbf{h}^{(l-1)}(j)\right) \tag{3}$$

where α_{ij} is a softmax function:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{a}^T\left[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j\right]\right)\right)}{\sum_{k\in\mathcal{N}(i)}\exp\left(\text{LeakyReLU}\left(\vec{a}^T\left[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k\right]\right)\right)}. \tag{4}$$

Here $\vec{a}$ is a vector of learnable weights and $\|$ represents vector concatenation. The coefficient α_{ij} represents the importance of node i to node j.

2.3 GraphSAGE (SAmple and aggreGatE)

In many graph problems, the number of edges can be large resulting in large sizes for neighborhoods $\mathcal{N}_i$. Methods such as GCN are also mainly transductive, meaning they do not perform well on unseen data. E.g. they do not cope well with graphs or nodes not seen in training. GraphSAGE [6] introduced sampling to reduce data complexity an improve generality. The update function of GraphSAGE is the similar to GCN, except that the neighbourhood is a smaller set randomly drawn from the real neighbours $\mathcal{N}_i$. GraphSAGE also performs better on unseen nodes.

3 MP-GNN: NEURAL NETWORKS FOR MOONLIGHTING PROTEIN PREDICTIONS

In this section, we first describe the label features we use to enhance model performance, then the construction of the PPI networks and finally the design of neural networks and their training process.

3.1 Sequence derived physiochemical features

Physical and chemical properties of proteins are determined by their amino acid sequences. Thus features derived from these sequences are called structure derived physiochemical features, or *physiochemical features* in brief. Formally, a physiochemical feature can be seen as a function $f : \mathcal{P} \to \mathbb{R}^n$ – embedding any sequence in the set of possible sequences $\mathcal{P}$ into a Euclidean space.

Such physiochemical features have been the basis of using machine learning methods like KNN, RF, and SVM [2, 9, 25]. In this paper we make use of use the following physiochemical features that were introduced in [2].

Unpublished working draft. Not for distribution.

3.1.1 quasi-sequence-order (QSOrder). The quasi-sequence-order (QSOrder) feature, denoted as $f(k)$, is defined based on the normalized occurrence of amino acid type k (where k ranges from 1 to 50) The formula for $f(k)$ is given by:

$$f(k) = \begin{cases} \frac{f_k}{\sum_{i=1}^{20} f_i + \omega \sum_{d=1}^{30} \tau_d} & \text{if } 1 \leq k \leq 20 \\ \frac{\omega \tau_{k-20}}{\sum_{i=1}^{20} f_i + \omega \sum_{d=1}^{30} \tau_d} & \text{if } 21 \leq k \leq 50 \end{cases} \tag{5}$$

In this formula, f_k is the frequency of amino acid type k. ω is a weighting factor and is given as 0.1 in our setting, and τ_k is the Sequence-order coupling number, defined in [2]. This descriptor provides a combination of sequence composition and sequence order information.

3.1.2 Amino Acid Composition (AAC). Amino Acid Composition (AAC) refers to the proportion of each amino acid in a protein sequence, for the 20 sommon amino acids. Specifically, the AAC for a protein sequence p is a vector in $\mathbb{R}^{20}$, in which the ith entry of the AAC vector x is calculated as:

$$x_i = \frac{n_i}{\sum_{j=1}^{20} n_j} \tag{6}$$

where n_i represents the number of occurrences of amino acid i.

3.2 Full PPI networks and Global MP-GNN model

3.2.1 Construction of Full PPI networks. Each protein in the graph $G = (P, A)$ is represented as a node included in the set P. For each pair of proteins P_i and P_j, a confidence value $e_{i,j}$ represents the likelihood of interaction (known from existing experimental or statistical data). We add an edge $\mathbf{A}_{i,j} = 1$ if the confidence $e_{i,j} > \theta$, where θ is a predetermined threshold. When they do not interact or the edge comes with a confidence lower than the threshold, we set $\mathbf{A}_{i,j} = 0$.

3.2.2 Global MP-GNN design and pipeline. Figure 1(a) illustrates the pipeline of the global MP-GNN framework. The GNN model is a trained model with a width of 256 and a depth of 3.

Given a set of query proteins, we pass them through the following process: (i) Construct the partially labeled full PPI network $G = (P, \mathbf{A}, y)$ for any species as described above, each protein $p \in P$ can be classified as either an MP, with a label $y_p = 1$, or a non-MP, with a label $y_p = 0$. Only a subset of nodes, $P_L \subseteq P$, are labeled, and the unlabeled nodes are denoted by P_U; (ii) extract the physiochemical feature x_p of each protein $p \in P$, and construct the node feature matrix $\mathbf{X}$; (iii) apply the GNN model to predict label y_p for each protein p based on $\mathbf{X}$ and $\mathbf{A}$.

3.3 Ego Networks and Local MP-GNN model

3.3.1 Construction of local PPI networks. This approach uses the local neighborhood of a query protein to make the prediction. Given each protein we use its K closest interacting partners, to construct the local neighborhood graph called the ego network. We recommend setting the graph size between 20 and 100 closest interacting partners to balance the representation of interacting patterns against the inclusion of noise and uncertain edges. We assume all interactions are bidirectional. Like the full-network method, edges can be labeled with weights, and we apply a threshold θ to filter edges. For all pairs i, j, we set $\mathbf{A}_{i,j} = 1$ if the confidence is greater than θ, and 0 if unconnected or below θ. Given each protein we use its K closest interacting partners, to construct the local neighborhood graph called the ego network. We recommend setting the graph size between 20 and 100 closest interacting partners to balance the representation of interacting patterns against the inclusion of noise and uncertain edges. We assume all interactions are bidirectional. Like the full-network method, edges can be labeled with weights, and we apply a threshold θ to filter edges. For all pairs i, j, we set $\mathbf{A}_{i,j} = 1$ if the confidence is greater than θ, and 0 if unconnected or below θ.

3.3.2 Local MP-GNN design and pipeline. An illustration of the pipeline is presented in Figure 1(b). We pass the graph into the GNN layer first, with a width of 256, and then to the graph pooling layer. We repeat this process three times. Graph pooling involves a form of coarsesning of the graph, where groups of nodes are aggregated into single nodes resulting in a more compact representation of the graph. There are several forms of graph pooling that can be used to coarsen graphs [36].

Given a query protein p, we pass it through the following process: (i) extract its *ego-network* with its closest K neighbours from the STRING database, as described in Section 4.1.3; (ii) construct the ego-network $\mathbf{G_p} = (V_p, \mathbf{A_p})$ of p as described above, extract the physiochemical feature $x_{p'}$ of each protein $p' \in V_p$, and construct the node feature matrix $\mathbf{X}$; (iv) apply GNN model $GNN : (\mathbf{X_p}, \mathbf{A_p}) \rightarrow \{0, 1\}$ to predict whether the query protein is an MP.

3.4 Training

For each graph convolutional layer—GAT, GCN, and GraphSAGE—we adopted a three-layer structure with a width of 256. We employed *EarlyStopping* as regularization setting a patience of 30. The learning rate was dynamically adjusted, starting at an initial rate of 0.001 and halved after 10 epochs without performance gains. We used the Adam [15] optimizer, and the Binary Cross Entropy (BCE) loss [23] as the loss function:

$$\mathcal{L}(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i) \right] \tag{7}$$

For model selection, we first randomized the data to prevent model bias. Then, we set aside 15% of the labeled data as an independent test set. The remaining labeled data underwent 5-fold cross-validation. In each fold, we trained the model for a maximum of 100 epochs on the training set, while monitoring the model's performance with EarlyStopper on the validation set. Finally, we selected the top three models based on their performance on the validation dataset to form a soft-voting ensemble model. We then tested the performance of the ensemble model on the test set and used its performance to evaluate the model performance.

Then, to further enhance the model performance and address the challenge of sparse labeling of MPs and non-MPs, we applied *semi-supervised learning* [18]. Specifically, we utilize the *pseudo-label* method[21]. We first trained the models on the labeled data for 50 epochs, then we employed the models to predict unlabeled nodes and selected high-confidence predictions, with a threshold of 0.9, as *pseudo-samples*. Then, we augmented the labeled dataset

Unpublished working draft. Not for distribution.

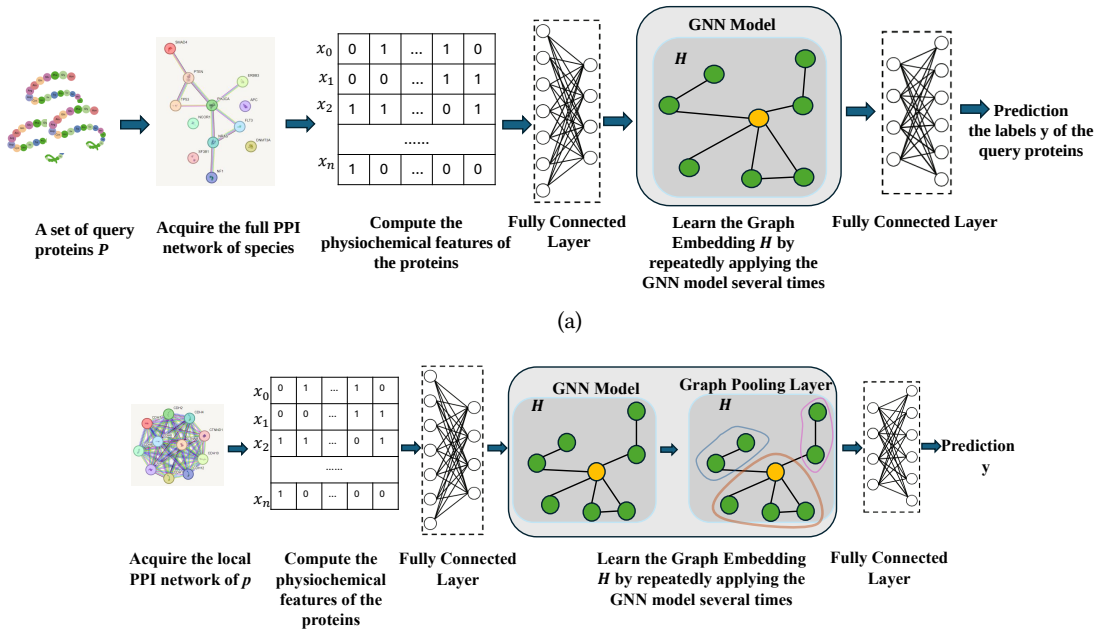

Figure 1: (a) Global MP-GNN framework. Operating on the full PPI network of a species (e.g. Homo sapiens). The GNN has three layers and a width of 256. The GNN layers are repeated 3 times. (b) Local MP-GNN framework. Operating on the local neighborhood of a node. The GNN and pooling layers are repeated 3 times.

with this *pseudo-samples* and continued training for an additional 30 epochs. This process was repeated for 10 times, and we recorded the final model performance on the test sets.

4 EXPERIMENTS

We carried out experiments in classifying moonlighting and non moonlighting proteins. We compare the various graph neural network ideas with conventional machine learning methods of KNN, SVM, RF and DNN found in [9, 25].

4.1 Dataset and network construction

This section describes the protein data used in experiments and the construction of PPI networks.

4.1.1 Moonlighting Proteins. The experiments employed the same benchmark dataset as [25]. It consists of 351 proteins, including 215 moonlighting proteins (MPs) and 136 non-moonlighting proteins (non-MPs)[1]. These proteins represent a wide array of organisms, including humans, E. coli, yeast, rats, fruit flies, and thale cress. The statistics of the data are given in Table 1. The MP data were sourced from the MoonProt database [20]. To minimize redundancy and ensure data quality, proteins with more than 40% sequence similarity using CD-hit [4] are excluded. The non-MP examples were the same as those used in MPFit [14].

[1]Note that it is hard to guarantee that any particular protein is definitely non-moonlighting, however, these examples are widely believed to be singe-function.

Organism	MP Sample Size	nonMP Sample Size
Yeast	23	33
Arath	5	0
Ecoli	24	16
Homo Sapiens	57	48
Mus musculus	22	39
Other	84	0

Table 1: Statistics for the data. The first column indicates the organism name, the last two columns indicate the contained MPs and non-MPs of the corresponding species in Shirafkan et al. [25]'s dataset.

4.1.2 Global PPI Networks. To construct the global PPI networks, we acquired the full networks specifically for *homo sapiens*, since only the *homo sapiens'* MP and non-MP data are balanced and large enough($\sim$ 50). As of the latest update, the network encompasses 19,622 nodes, with interactions assigned a confidence value ranging from 0 to 1000. To balance data quality with graph size, we applied a confidence threshold of 250, resulting in a refined graph that contains 13,524,114 edges. Within this network, all 57 MPs and 48 non-MPs of *homo sapiens* were accurately matched.

4.1.3 Local PPI Networks. We downloaded the Protein-protein Interaction (PPI) information from the STRING database, which provides comprehensive protein interaction data for each species [27].

Unpublished working draft. Not for distribution.

To construct the local PPI networks, we queried the closest 10-100 interacting partners for each MP and non-MP. These interactions could be either physical or functional, and are assigned a score between 0 to 1 based on various types of evidence. To maintain interaction, we filtered these data by a threshold of 0.7. Out of the 215 MPs listed in Shirafkan et al. [25], 174 were found within the STRING database, accounting for 80.9% of the MPs. All example non-MPs were also successfully located within the database. Consequently, our finalized dataset comprised 310 proteins, including 174 MPs and 136 non-MPs. For each protein, we obtained PPI networks of various sizes, ranging from their 20 closest neighbors up to 100, to facilitate a thorough analysis.

Table 2: The experimental results of different combinations of naive graph convolutional layers node features in the global MP-GNN framework, and the corresponding performances of the baseline models. The table shows that the global MPGNN models consistently outperformed previous methods; specifically, the graphSAGE model in the semi-supervised learning setting achieved the top performance in all metrics, significantly outperforming previous methods by about 10-20% .

Feature	Model	ROC-AUC	Prec.	F1	Rec.	Acc.
QSOrder	**GAT**	71.4	78.4	77.6	76.8	78.2
	GCN	71.4	78.4	77.5	76.6	78.1
	SAGE	79.7	85.1	83.9	82.7	84.4
	GAT (SSL)	72.8	80.1	79.4	80.7	79.9
	GCN (SSL)	72.2	80.7	80.6	80.5	80.4
	SAGE (SSL)	80.9	88.4	87.6	**89.2**	87.6
	KNN	66.0	67.0	74.9	88.8	67.6
	SVM	68.9	74.2	70.9	79.4	70.5
	RF	64.8	69.4	67.9	79.0	66.0
	DNN	66.8	70.2	70.5	75.9	65.1
AAC	**GAT**	70.9	78.4	78.2	78.0	78.1
	GCN	72.5	71.6	71.1	70.6	71.9
	SAGE	78.6	81.3	80.9	**89.5**	81.3
	GAT (SSL)	72.1	74.3	73.6	72.9	73.8
	GCN (SSL)	72.3	74.6	74.1	73.6	74.5
	SAGE (SSL)	74.9	78.2	77.2	76.2	76.9
	KNN	65.8	73.8	67.0	89.0	67.6
	SVM	70.5	72.4	74.2	79.1	70.5
	RF	65.8	69.0	69.4	79.4	66.0
	DNN	65.5	67.4	70.2	76.0	65.1
AAC + QSOrder	**GAT**	73.5	79.3	78.5	77.7	78.5
	GCN	72.4	78.9	78.2	77.5	78.3
	SAGE	**81.2**	**85.3**	**84.2**	89.3	**85.4**
	GAT (SSL)	72.8	80.1	79.8	79.5	82.2
	GCN (SSL)	72.3	81.3	80.7	80.1	82.4
	SAGE (SSL)	**81.3**	**88.0**	**88.8**	87.2	**88.4**
	KNN	65.3	66.1	67.6	88.8	67.6
	SVM	69.9	73.7	70.0	79.4	70.0
	RF	66.8	70.1	67.1	78.6	67.1
	DNN	65.9	69.6	66.3	77.3	66.3

Table 2 and Figure 2 present the results of various GNN models, in local and global, the supervised and unsupervised settings, and a comparison with four top-performing baseline methods as identified in the works of Shirafkan et al. [25] and Hu and Du [9]: KNN, SVM, RF, and DNN, on the data of *Homo Sapiens* only. As seen in Figure 2, while some conventional methods do well in some metrics only the graph neural networks do well consistently in all the metrics.

Generally, semi-supervised learning (SSL) is found to improve the model performance by about 2-3%, for all feature types. GraphSAGE exhibits the best performance across all metrics, in both the supervised and semi-supervised settings, and its performance is maximized with the AAC+QSOrder feature combination. In the supervised setting, the GraphSAGE model exceeded the top-performing baseline model by 11.1% in accuracy, 4.9% in F-1 score, 8.8% in precision, and 7.4% in ROC-AUC; in the semi-supervised setting, the model performance was further improved by about 2-3%, achieving an accuracy of 88.4%, recall of 87.2%, F1-score of 88.0%, precision of 88.8%, and ROC-AUC of 81.3%.

The results of GAT and GCN models were consistently below GraphSAGE by about 5% but still exhibit improvements of about 1-5% in all metrics over the baseline methods. Similarly, when we applied SSL to them, these models were improved by about 2-3% across every metric.

For features, we found that the combination of AAC and QSOrder produces the best results. Using GraphSAGE as an example, its accuracy with QSOrder alone was 84.4%, recall was 82.7%, F-1 score was 83.9%, precision was 85.1%, and ROC-AUC was 79.7%; its accuracy with AAC alone was 81.3%, recall was 89.5%, F-1 score was 80.9%, precision was 81.3%, and ROC-AUC was 78.6%. However, when using the concatenation of AAC and QSOrder as the node feature, its accuracy increases to 85.4%, F-1 score to 84.2%, precision to 85.3%, and ROC-AUC to 81.2%. Only recall did not show improvement. Similar observations were observed across all models, including previous methods.

4.2 Local MP-GNN

4.2.1 Full Dataset Results. Table 3 presents the results for the local MP-GNN framework on various GNN variants on the full dataset, with various types of graph pooling layers. We can see that even the naive implementations of the GNN models have already slightly outperformed the baseline methods, and the combination of various graph pooling layers consistently further improved these models' performance.

The GraphSAGE model with SAGPooling layer, utilizing the concatenation of QSOrder and AAC achieved the best performance, with accuracy of 79.6%, recall of 78.0%, F1-score of 79.0%, precision of 80.0%, and ROC-AUC of 75.8%, respectively, surpassing the best baseline models by about 5% in accuracy and 8% in precision. Other GNN models, including GAT and GCN, also demonstrated comparable performance.

Among the graph pooling layers we tested [5], the Self-Attention Graph Pooling (SAGPooling) [19] stands out to be the most effective pooling layer. Notably, the combination of GAT with SAGPooling, utilizing the AAC+QSOrder feature combination, exhibits the best performance. When paired with the GraphSAGE model, it achieved the highest metrics, with an accuracy of 79.6%, recall of 78.0%, F-1

Unpublished working draft. Not for distribution.

Table 3: The experimental results of different combinations of Graph Pooling Layers, naive graph convolutional layers node features in the local MP-GNN framework, and the corresponding performances of the baseline models. The table shows that graph pooling layers consistently improve the model performances. Specifically, the SAGPooling layer exhibited the best effect, and the combination of GraphSAGE, AAC+QSOrder, and SAGPooling demonstrated the best accuracy, ROC-AUC, and Precision.

Feature	Model	ROC-AUC	Precision	F1-Score	Recall	Accuracy
QSOrder	GAT	72.8	75.4	74.5	73.6	74.9
	GCN	72.2	74.4	74.0	73.6	74.4
	SAGE	75.0	78.1	77.1	76.1	77.2
	GAT+SAGPooling	74.6	77.8	77.1	76.4	77.3
	GCN+SAGPooling	71.3	75.5	74.5	73.5	74.8
	SAGE+SAGPooling	75.6	79.9	78.9	77.9	79.0
	GAT+EdgePooling	74.9	78.1	76.7	75.3	76.7
	SAGE+EdgePooling	74.9	78.2	77.7	76.7	77.8
	GCN+EdgePooling	71.9	75.2	74.1	73.0	74.1
	GAT+ASAPooling	75.1	77.9	76.5	75.1	76.4
	GCN+ASAPooling	72.0	75.5	74.4	73.3	74.4
	SAGE+ASAPooling	75.7	79.2	78.2	77.2	78.5
	KNN	70.7	71.0	**78.9**	88.8	73.1
	SVM	73.5	76.1	77.7	79.4	74.2
	RF	72.2	74.9	76.8	79.0	73.0
	DNN	69.5	72.9	74.4	75.9	70.3
AAC	GAT	72.1	74.3	73.6	72.9	73.8
	GCN	72.3	74.6	74.1	73.6	74.5
	SAGE	74.9	78.2	77.2	76.2	76.9
	GAT+SAGPooling	74.6	77.8	77.1	76.4	77.3
	GCN+SAGPooling	73.6	77.4	76.2	75.0	76.4
	SAGE+SAGPooling	75.7	80.0	79.0	78.0	78.5
	GAT+EdgePooling	74.9	78.1	76.7	75.3	76.7
	GCN+EdgePooling	71.4	74.8	73.4	72.0	73.3
	SAGE+EdgePooling	75.0	78.0	77.2	76.4	77.5
	GAT+ASAPooling	73.8	76.7	75.6	74.5	75.6
	GCN+ASAPooling	71.9	75.5	73.9	72.3	73.9
	SAGE+ASAPooling	75.4	79.1	78.1	77.1	78.3
	KNN	71.1	71.5	**79.3**	**89.0**	73.6
	SVM	73.8	76.5	77.8	79.1	74.4
	RF	70.9	73.4	76.3	79.4	71.9
	DNN	68.6	72.1	74.0	76.0	69.8
AAC + QSOrder	GAT	73.5	75.5	74.5	73.5	74.6
	GCN	72.4	74.9	74.2	73.5	74.5
	SAGE	75.1	78.3	77.3	76.3	77.3
	GAT+SAGPooling	75.4	78.8	77.9	77.0	78.0
	GCN+SAGPooling	72.3	77.0	75.8	74.6	76.2
	SAGE+SAGPooling	**75.8**	**80.0**	79.0	78.0	**79.6**
	GAT+EdgePooling	73.9	77.1	75.7	74.3	75.7
	GCN+EdgePooling	71.3	74.7	73.3	71.9	73.3
	SAGE+EdgePooling	75.8	78.3	77.7	76.8	77.8
	GAT+ASAPooling	74.6	77.5	76.2	74.9	76.2
	GCN+ASAPooling	71.8	75.1	74.0	72.9	74.1
	SAGE+ASAPooling	75.6	79.3	78.3	77.3	78.6
	KNN	70.8	71.0	78.9	88.8	73.1
	SVM	73.7	76.3	77.8	79.4	74.4
	RF	72.4	75.3	76.9	78.6	73.1
	DNN	70.2	73.2	75.2	77.3	71.0

Unpublished working draft. Not for distribution.

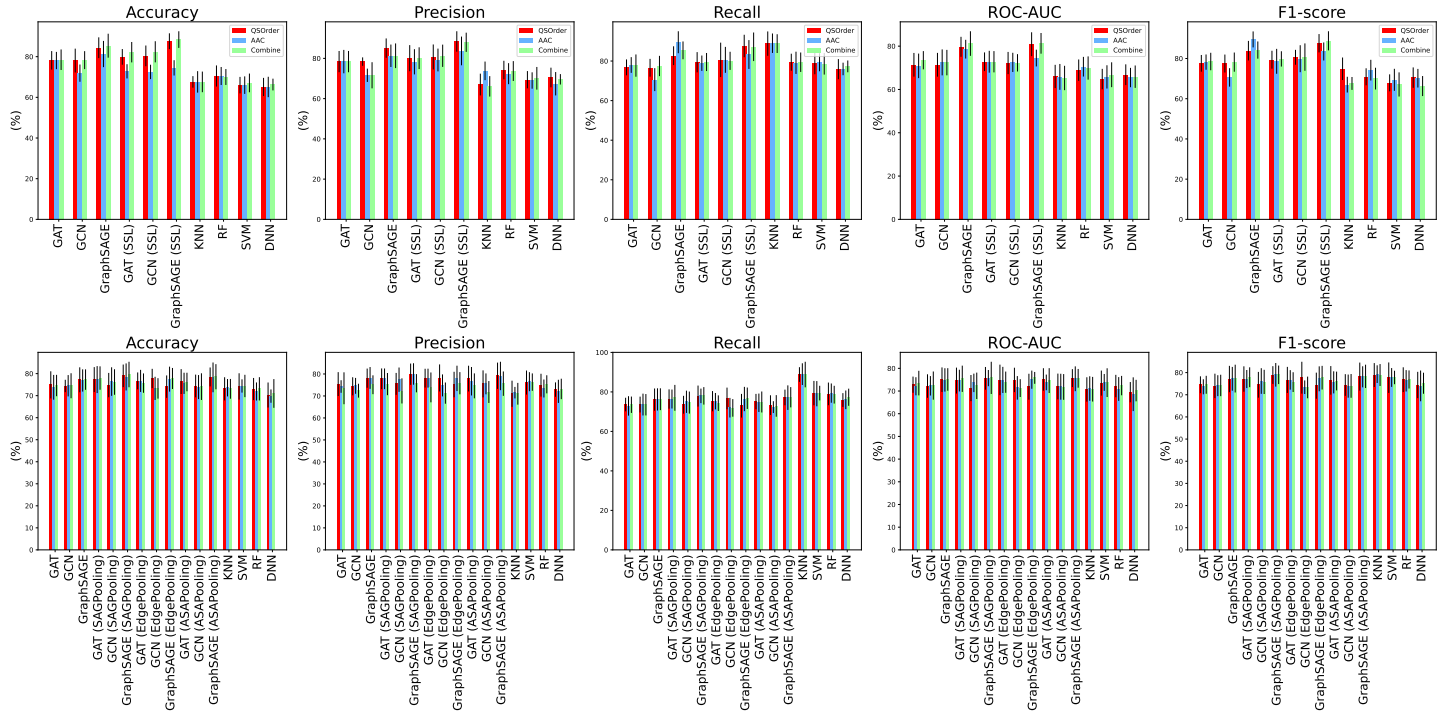

Figure 2: A comparison between all global MP-GNN models (top row) and local MP-GNN models with different graph pooling layers (bottom row) and the baseline models. From the plots, we can see that the global MPGNN model with GraphSAGE in the semi-supervised learning setting exhibited the top performance, significantly outperforming other (both global and local) MPGNNs and the baseline models

score of 79.0%, precision of 80.0%, and ROC-AUC of 75.8%. These results significantly surpassed those of previous methods.

Other pooling layers, such as EdgePooling and ASAPooling, also showed significant improvements with the GAT and GraphSAGE models. However, GCNs did not demonstrate significant enhancement when combined with any of the pooling layers.

4.2.2 The effect of Graph Size on the model performance. To test the effect of various neighborhood sizes on the model performance, we assessed the model's performance across neighborhood graph sizes ranging from 10 to 100. The accuracy trends are depicted in Figure 3.

We can see that the model performance peaks at a size of 50, and marginally declines when the graph size is set to 30. For smaller graph sizes, performance significantly decreases by about 2% to 3%. Conversely, increasing the graph size to 60, and then further to 100, results in a gradual reduction in model performance. Thus, empirically, neighborhood size of 50 appears a good practical choice.

This observation likely underscores the trade-off between graph noise and information content. With small graph sizes, there is insufficient encapsulation of local patterns and interactions, leading to underperformance. Conversely, larger neihborhood sizes introduce a higher level of noise, detracting from model effectiveness.

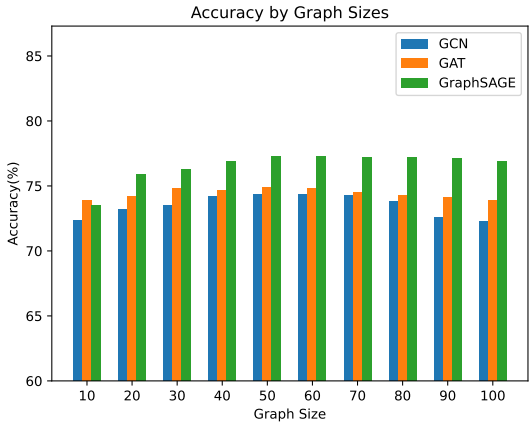

Figure 3: The accuracy of the Local GNN models on neighborhood graphs of sizes 10-100. We also observed similar trends for other metrics, including F-1, recall, precision, and AUC-ROC. We can see that the model performance is approximately maximized when the graph size is between 50-60. This trend is consistent across different GNN models

Unpublished working draft. Not for distribution.

5 DISCUSSION

We see that GNN models successfully combine the PPI data and physicochemical features of proteins and could distinguish MPs from non-MPs.

Across the experiments, global MP-GNN exhibited stronger performance than the local MP-GNN method. We have also run the local MP-GNN models using only on the data of *homo sapiens*, but found that the performance did not match that of the global MP-GNN.

For complexity, each single local MP-GNN with a size of 50 occupied space less than 500KB, and each training and evaluation took between 0.5-2 hours to run on a standard 3070 machine with 1 CUDA available. The full PPI network of homo sapiens was about 601MB, and the whole training and evaluation took between 15-20 hours to complete.

The accuracy difference between the global and local MP-GNNs suggests that the position of MPs in the broader graph is significant. In further work, the behaviors of MPs and non-MPs in the global and local PPI networks can be further explored. We evaluated the clustering coefficients of the local PPI graphs of the 174 MPs and 136 non-MPs. As an example, we applied statistical analysis to show that MPs (0.38 ± 0.21) exhibited higher mean clustering coefficients than non-MPs (0.27 ± 0.24), with $p < 0.01$. This suggests that the MPs are more likely to form cliques with their neighbors in the PPI networks, while non-MPs are less likely to do so.

That the GraphSAGE model outperformed the GAT model is possibly an influence of noise and small size of labelled samples for the problem. In such noisy scenarios, with possibly incorrect labels, GAT is known to perform poorly [24]. The performance did not change appreciably when we tried types of attention mechanisms – including cardinality preservation [34] and regularization [24]. This circumstance may change as more moonlighting proteins labels (and non-moonlighting proteins) become known (possibly with help of this and similar works).

We investigated the edge weights in GAT models of local PPI networks. We found that the GAT model puts large weights on the self loops – presented as larger values along the diagonal. A couple of examples of this phenomenon is shown in Figure 4. This pattern suggests that the model puts an emphasis on the presevation of individual features of nodes in the neighborhood. Thus, it would appear that this distribution of physiochemical features within the neighborhoods is important to the GAT neural network.

6 CONCLUSION

There are several fronts on which this line of research can make progress. We have noted above that the data available on protein properties is relatively sparse. This is a somewhat chicken-and-egg problem. We hope that methods such as these will aid in identifying new moonlighting proteins and more labels will allow for better, more accurate models.

On the computational and statistical side, *Positive unlabelled learning* – a successor of semi-supervised learning used here – is an interesting topic in current development that can push the state of the art in this area. As can newer generations of GNNs and feature engineering.

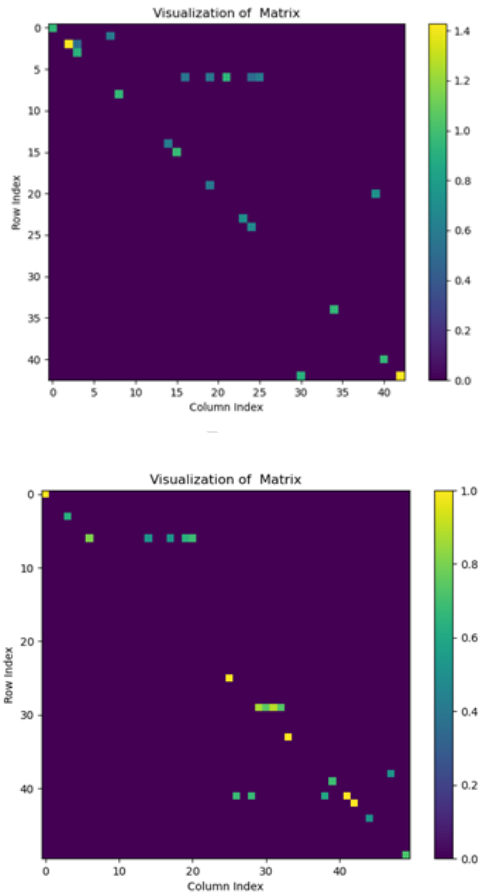

Figure 4: A comparison between the Graph Attention Layer weights of MPs(top) and nonMPs(bottom) of the attention weights of two random samples. The figures were filtered with a threshold of 0.4, highlighting only those with importance. We can see that most of the important edges are centred along the diagonal line.

REFERENCES

[1] Charles E Chapple, Benoit Robisson, Lionel Spinelli, Céline Guien, Emmanuelle Becker, and Christine Brun. 2015. Extreme multifunctional proteins identified from a human protein interaction network. *Nature communications* 6, 1 (2015), 7412.

[2] Chao Chen, Li-Xuan Chen, Xiao-Yong Zou, and Pei-Xiang Cai. 2008. Predicting protein structural class based on multi-features fusion. *Journal of theoretical biology* 253, 2 (2008), 388–392.

[3] Luis Franco-Serrano, David Sánchez-Redondo, Araceli Nájar-García, Sergio Hernández, Isaac Amela, Josep Antoni Perez-Pons, Jaume Piñol, Angel Mozo-Villarias, Juan Cedano, and Enrique Querol. 2021. Pathogen moonlighting proteins: from ancestral key metabolic enzymes to virulence factors. *Microorganisms* 9, 6 (2021), 1300.

[4] Limin Fu, Beifang Niu, Zhengwei Zhu, Sitao Wu, and Weizhong Li. 2012. CD-HIT: accelerated for clustering the next-generation sequencing data. *Bioinformatics* 28, 23 (2012), 3150–3152.

[5] Daniele Grattarola, Daniele Zambon, Filippo Maria Bianchi, and Cesare Alippi. 2022. Understanding pooling in graph neural networks. *IEEE transactions on neural networks and learning systems* (2022).

[6] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30

(2017).
[7] Sergio Hernández, Alejandra Calvo, Gabriela Ferragut, Luís Franco, Antoni Hermoso, Isaac Amela, Antonio Gómez, Enrique Querol, and Juan Cedano. 2014. Can bioinformatics help in the identification of moonlighting proteins? *Biochemical Society Transactions* 42, 6 (2014), 1692–1697.
[8] Sergio Hernández, Luís Franco, Alejandra Calvo, Gabriela Ferragut, Antoni Hermoso, Isaac Amela, Antonio Gómez, Enrique Querol, and Juan Cedano. 2015. Bioinformatics and moonlighting proteins. *Frontiers in bioengineering and biotechnology* 3 (2015), 90.
[9] Jing Hu and Yihang Du. 2023. Predicting Moonlighting Proteins from Protein Sequence. (2023).
[10] Constance J Jeffery. 2018. Protein moonlighting: what is it, and why is it important? *Philosophical transactions of the Royal Society B: biological sciences* 373, 1738 (2018), 20160523.
[11] Kanchan Jha, Sriparna Saha, and Hiteshi Singh. 2022. Prediction of protein–protein interaction using graph neural networks. *Scientific Reports* 12, 1 (2022), 8360.
[12] Ishita Khan, Yuqian Chen, Tiange Dong, Xioawei Hong, Rikiya Takeuchi, Hirotada Mori, and Daisuke Kihara. 2014. Genome-scale identification and characterization of moonlighting proteins. *Biology direct* 9, 1 (2014), 1–29.
[13] Ishita K Khan, Mansurul Bhuiyan, and Daisuke Kihara. 2017. DextMP: deep dive into text for predicting moonlighting proteins. *Bioinformatics* 33, 14 (2017), i83–i91.
[14] Ishita K Khan and Daisuke Kihara. 2016. Genome-scale prediction of moonlighting proteins using diverse protein association information. *Bioinformatics* 32, 15 (2016), 2281–2288.
[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[16] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[17] Boqiao Lai and Jinbo Xu. 2022. Accurate protein function prediction via graph attention networks with predicted structure information. *Briefings in Bioinformatics* 23, 1 (2022), bbab502.
[18] Semi-Supervised Learning. 2006. Semi-supervised learning. *CSZ2006. html* 5 (2006).
[19] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-attention graph pooling. In *International conference on machine learning*. PMLR, 3734–3743.
[20] Mathew Mani, Chang Chen, Vaishak Amblee, Haipeng Liu, Tanu Mathur, Grant Zwicke, Shadi Zabad, Bansi Patel, Jagravi Thakkar, and Constance J Jeffery. 2015. MoonProt: a database for proteins that are known to moonlight. *Nucleic acids research* 43, D1 (2015), D277–D282.
[21] Nitin Namdeo Pise and Parag Kulkarni. 2008. A survey of semi-supervised learning methods. In *2008 International conference on computational intelligence and security*, Vol. 2. IEEE, 30–34.
[22] Jason A Reuter, Damek V Spacek, and Michael P Snyder. 2015. High-throughput sequencing technologies. *Molecular cell* 58, 4 (2015), 586–597.
[23] Usha Ruby and Vamsidhar Yendapalli. 2020. Binary cross entropy with deep learning technique for image classification. *Int. J. Adv. Trends Comput. Sci. Eng* 9, 10 (2020).
[24] Uday Shankar Shanthamallu, Jayaraman J Thiagarajan, and Andreas Spanias. 2020. A regularized attention mechanism for graph attention networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 3372–3376.
[25] Farshid Shirafkan, Sajjad Gharaghani, Karim Rahimian, Reza Hasan Sajedi, and Javad Zahiri. 2021. Moonlighting protein prediction using physico-chemical and evolutional properties via machine learning methods. *BMC bioinformatics* 22, 1 (2021), 1–14.
[26] Nadia Singh and Needhi Bhalla. 2020. Moonlighting proteins. *Annual review of genetics* 54 (2020), 265–285.
[27] Damian Szklarczyk, Annika L Gable, Katerina C Nastou, David Lyon, Rebecca Kirsch, Sampo Pyysalo, Nadezhda T Doncheva, Marc Legeay, Tao Fang, Peer Bork, et al. 2021. The STRING database in 2021: customizable protein–protein networks, and functional characterization of user-uploaded gene/measurement sets. *Nucleic acids research* 49, D1 (2021), D605–D612.
[28] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[29] Graeme Wistow and Joram Piatigorsky. 1987. Recruitment of enzymes as lens structural proteins. *Science* 236, 4808 (1987), 1554–1556.
[30] Graeme J Wistow and Joram Piatigorsky. 1988. Lens crystallins: the evolution and expression of proteins for a highly specialized tissue. *Annual review of biochemistry* 57, 1 (1988), 479–504.
[31] Hai-Cheng Yi, Zhu-Hong You, De-Shuang Huang, and Chee Keong Kwoh. 2022. Graph representation learning in bioinformatics: trends, methods and applications. *Briefings in Bioinformatics* 23, 1 (2022), bbab340.
[32] Ronghui You, Shuwei Yao, Hiroshi Mamitsuka, and Shanfeng Zhu. 2021. DeepGraphGO: graph neural network for large-scale, multispecies protein function prediction. *Bioinformatics* 37, Supplement_1 (2021), i262–i271.
[33] Liyuan Zhang, Yongquan Jiang, and Yan Yang. 2023. Gnngo3d: Protein function prediction based on 3d structure and functional hierarchy learning. *IEEE Transactions on Knowledge and Data Engineering* (2023).
[34] Shuo Zhang and Lei Xie. 2020. Improving attention mechanism in graph neural networks via cardinality preservation. In *IJCAI: proceedings of the conference*, Vol. 2020. NIH Public Access, 1395.
[35] Chenguang Zhao, Tong Liu, and Zheng Wang. 2022. PANDA2: protein function prediction using graph neural networks. *NAR genomics and bioinformatics* 4, 1 (2022), lqac004.
[36] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI open* 1 (2020), 57–81.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009

Unpublished working draft. Not for distribution.