Chapter 25
Graph Neural Networks in Predicting Protein Function and Interactions

Anowarul Kabir and Amarda Shehu

Abstract Graph Neural Networks (GNNs) are becoming increasingly popular and powerful tools in molecular modeling research due to their ability to operate over non-Euclidean data, such as graphs. Because of their ability to embed both the inherent structure and preserve the semantic information in a graph, GNNs are advancing diverse molecular structure-function studies. In this chapter, we focus on GNN-aided studies that bring together one or more protein-centric sources of data with the goal of elucidating protein function. We provide a short survey on GNNs and their most successful, recent variants designed to tackle the related problems of predicting the biological function and molecular interactions of protein molecules. We review the latest methodological advances, discoveries, as well as open challenges promising to spur further research.

25.1 From Protein Interactions to Function: An Introduction

Molecular biology is now reaping the benefits of big data, as rapidly advancing high-throughput, automated wet-laboratory protocols have resulted in a vast amount of biological sequence, expression, interactions, and structure data (Stark, 2006; Zoete et al, 2011; Finn et al, 2013; Sterling and Irwin, 2015; Dana et al, 2018; Doncheva et al, 2018). Since functional characterization has lagged behind, we now have millions of protein products in databases for which no functional information is readily available; that is, we do not know what many of the proteins in our cells do (Gligorijevic et al, 2020).

Anowarul Kabir
Department of Computer Science, George Mason University, e-mail: akabir4@gmu.edu

Amarda Shehu
Department of Computer Science, George Mason University, e-mail: amarda@gmu.edu

Answering the question of what function a protein molecule performs is key not only to understanding our biology and protein-centric disorders, but also to advancing protein-targeted therapies. Hence, this question remains the driver of much wet- and dry-laboratory research in molecular biology (Radivojac et al, 2013; Jiang et al, 2016). Answering it can take many forms based on the detail sought or possible. The highest amount of detail provides an answer to the question by directly exposing the other molecules with which a target protein interacts in the cell, thus revealing what a protein does by elucidating the molecular partners to which it binds.

In this brief survey, we focus on how graph neural networks (GNNs) are advancing our ability to answer this question *in silico*. This chapter is organized as follows: First, a brief historical overview is provided, so that the reader understands the evolution of ideas and data that have made possible the application of machine learning to the problem of protein function prediction. Then, a brief overview of the (shallow) models prior to GNNs is provided. The rest of the survey is devoted to the GNN-based formulation of this question, a summary of state-of-the-art (SOTA) GNN-based methods, with a few selected methods highlighted where relevant, and an exposition of remaining challenges and potential ways forward via GNNs.

25.1.1 Enter Stage Left: Protein-Protein Interaction Networks

Historically, the earliest methods devised for protein function prediction related protein sequence similarity to protein function similarity. This led to important discoveries until remote homologs were identified, which are proteins with low sequence similarity but highly similar three-dimensional/tertiary structure and function. So methods evolved to utilize tertiary structure, but their applicability was limited, as determination of tertiary structure was and remains a laborious process. Other methods utilized patterns in gene expression data to infer interacting proteins, based on the insight that proteins interacting with one another need foremost to be expressed in the cell at the same time.

With the development of high-throughput technologies, such as two-hybrid analysis for the yeast protein interactome (Ito et al, 2001), tandem-affinity purification and mass spectrometry (TAP-MS) (Gavin et al, 2002) for characterizing multi-protein complexes and protein-protein associations (Huang et al, 2016a), high-throughput mass spectrometric protein complex identification (HMS-PCI) (Ho et al, 2002), co-immunoprecipitation coupled to mass spectrometry (Foltman and Sanchez-Diaz, 2016), protein-protein interaction (PPI) data suddenly became available, and in large amounts. PPI networks, with edges denoting interacting protein nodes, of many species, such as human, yeast, mouse, and others, suddenly became available to researchers. PPI networks, as small as a few nodes or as large as tens of thousands of nodes, gave a boost to machine learning methods and improved the performance of shallow models. Surveys such as Ref. (Shehu et al, 2016) provide a detailed history of the evolution of protein function prediction methods as different sources of wet-laboratory data became available to computational biologists.

25.1.2 Problem Formulation(s), Assumptions, and Noise: A Historical Perspective

A natural question arises. If we have access to PPI data, then what else remains to predict with regards to protein function? Despite significant progress, the reality remains that there are many unmapped PPIs. This is formally known as the *link prediction* problem. For various reasons, PPI networks are incomplete. They entirely miss information on a protein, or they may contain incomplete information on a protein. In particular, we now know that PPIs suffer from high type-I error, type-II error, and low inclusion (Luo et al, 2015; Byron and Vestergaard, 2015). The total number of PPI links that are experimentally determined is still moderate (Han et al, 2005). PPI data are inherently noisy as experimental methods often produce false-positive results (Hashemifar et al, 2018). Therefore, predicting protein function computationally remains an essential task.

The problem of protein function prediction is often formulated as that of link prediction, that is, *predicting whether or not there exists a connection between two nodes in a given PPI network*. While link prediction methods connect proteins on the basis of biological or network-based similarity, researchers report that interacting proteins are not necessarily similar and similar proteins do not necessarily interact (Kovács et al, 2019).

As indicated above, information on protein function can be provided at different levels of detail. There are several widely-used protein function annotation schemes, including the Gene Ontology (Lovell et al, 2003) (GO) Consortium, the Kyoto Encyclopedia of Genes and Genomes (Wang and Dunbrack, 2003) (KEGG), the Enzyme Commission (Rhodes, 2010) (EC) numbers, the Human Phenotype Ontology (Robinson et al, 2008), and others. It is beyond the scope of this paper to provide an explanation of these ontologies. However, we emphasize that the most popular one remains the GO annotation, which classifies proteins into hierarchically-related functional classes organized into 3 different ontologies: Molecular Function (MF), Biological Process (BP), and Cellular Component (CC), to describe different aspects of protein functions. Systematic benchmarking efforts via the Critical Assessment of Functional Annotation (CAFA) community-wide experiments (Radivojac et al, 2013; Jiang et al, 2016; Zhou et al, 2019b) and MouseFunc (Peña-Castillo et al, 2008) have been central to the automation of protein function annotation and rigorous assessment of devised methodologies.

25.1.3 Shallow Machine Learning Models over the Years

Many shallow machine learning approaches have been developed over the years. Xue-Wen and Mei propose a domain-based random forest of decision trees to infer protein interactions on the *Saccharomyces cerevisiae* dataset (Chen and Liu, 2005). Shinsuke *et al.* apply multiple support vector machines (SVMs) for predicting interactions between pairs of yeast proteins and pairs of human proteins by increasing more negative pairs than positives (Dohkan et al, 2006). Fiona *et al.* assess naïve bayes (NB), multi-layer perceptron (MLP) and k-nearest neighbour (KNN) methods on diverse, large-scale functional data to infer pairwise (PW) and module-based (MB) interaction networks (Browne et al, 2007). PRED_PPI provides a server developed on SVM for predicting PPIs in five organisms, such as humans, yeast, *Drosophila*, *Escherichia coli*, and *Caenorhabditis elegans* (Guo et al, 2010). Xiaotong and Xue-wen integrate features extracted from microarray expression measurements, GO labels and orthologous scores, and apply a tree-augmented NB classifier for human PPI predictions from model organisms (Lin and Chen, 2012). Zhu-Hong *et al.* propose a multi-scale local descriptor feature representation scheme to extract features from a protein sequence and use random forest (You et al, 2015a). Zhu-Hong *et al.* propose to apply SVM on a matrix-based representation of protein sequence, which fully considers the sequence order and dipeptide information of the protein primary sequence to detect PPIs (You et al, 2015b).

Although many advances were made by shallow models, as summarized in Table 25.1, the problem of protein function prediction is still a long way from being solved. Shallow machine learning methods depend greatly on feature extraction and feature computation, which hinder performance. The task of feature engineering, particularly when integrating different sources of data (sequence, expression, interactions) is complex, laborious, and ultimately limited by human creativity and domain-specific understanding of what may be determinants of protein function. In particular, feature-based shallow models cannot fully incorporate the rich, local and distal topological information present in one or more PPI networks. These reasons have prompted researchers to investigate GNNs for protein function prediction.

25.1.4 Enter Stage Right: Graph Neural Networks

This section first relates a general formulation of a GNN and forsakes detail in the interest of space, assuming readers are already somewhat familiar with GNNs. The rest of the section focuses on three task-specific formulations that allow leveraging GNNs for protein function prediction.

25.1.4.1 Preliminaries

Assume an undirected and unweighted molecular-interaction graph, i.e., a PPI network, is represented by $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ and $\mathscr{E}$ denote the set of vertices representing proteins and the edges indicating interactions among proteins, respectively. Let the i-th protein be represented as an m-dimensional feature vector; that is, $p_i \in \mathbb{R}^m$. The objective of a GNN is to learn an embedding, h_i, using the message passing protocol which essentially aggregates and transforms neighboring information to update the current node's vector representation. Assuming f and g are two

Table 25.1: Summary of performance of shallow models as reported in (Chen and Liu, 2005; Guo et al, 2010; Lin and Chen, 2012; You et al, 2015a,b)

Literature	Model	Dataset	Sensitivity (%)	Specificity (%)	Accuracy (%)
Chen and Liu (2005)	RF	*Saccharomyces cerevisiae*	79.78	64.38	NA*
Guo et al (2010)	SVM	Human	89.17	92.17	90.67
		Yeast	88.17	89.81	88.99
		Drosophila	99.53	80.65	90.09
		Escherichia coli	95.11	90.35	92.73
		Caenorhabditis elegans	96.46	98.55	97.51
Lin and Chen (2012)	Tree-Augmented Naïve Bayes (TAN)	Human	88	70	NA*
You et al (2015a)	RF	*Saccharomyces cerevisiae*	94.34	NA*	94.72
You et al (2015b)	SVM	*Saccharomyces cerevisiae*	85.74	94.37	90.06

* Not available

parametric functions that compute the embedding and output considering a single protein, following (Scarselli et al, 2008), we formulate follows:

$$h_i = f(p_i, p_{e[i]}, p_{ne[i]}, h_{ne[i]}) \tag{25.1}$$

$$o_i = g(h_i, p_i) \tag{25.2}$$

where p_i, $p_{e[i]}$, $p_{ne[i]}$ and $h_{ne[i]}$ denote the feature representation of the i-th protein, features of all connected edges to the i-th protein, neighboring proteins' features and embeddings of neighborhood proteins of the i-th protein, respectively.

Let us now consider $|\mathscr{V}| = n$ proteins. All proteins are represented as a matrix, $P \in \mathbb{R}^{n\times m}$. The adjacency matrix $A \in \mathbb{R}^{n\times n}$ encodes the connectivity of the proteins; namely, $A_{i,j}$ indicates whether or not there exists a link between proteins i and j. Enforcing the self-loops with each protein, the updated adjacent matrix is $\tilde{A} = A + I$. The degree diagonal matrix, D, can then be defined, such that $D_{i,i} = \sum_{j=1}^{n} \tilde{A}_{i,j}$. From there, one can compute the symmetric Laplacian matrix $L = D - \tilde{A}$. Finally, one can then formulate the following iterative process:

$$H^{t+1} = F(H^t, P||A||L||X) \tag{25.3}$$

$$O = G(H, P||A||L||X) \tag{25.4}$$

where H^t denotes t-th iteration of H, $(\cdot||\cdot)$ indicates the aggregation operation based on the task at hand, and O is the final stacked output.

25.1.4.2 GNNs for Representation Learning

We now want to encode complex high-dimensional information, such as a protein, P, or a biological interaction, A, or an interaction network, $\mathscr{G}$, into low-dimensional embeddings, Z, by capturing linearity and non-linearity among nodes and edges. In principle, the representation should contain all the information for downstream machine learning tasks, such as link prediction, protein classification, protein cluster analysis, interaction prediction, etc.

Suppose we want to learn a graph embedding, Z, from the network $\mathscr{G}$. A graph auto-encoder neural network (Kipf and Welling, 2016) can be applied to learn Z:

$$Z = GNN(P, A; \theta_{gnn}) \tag{25.5}$$

where θ_{gnn} denotes GNN (encoder)-specific learnable parameters.

25.1.4.3 GNNs for the Link Prediction Problem

Given two proteins, we want to predict if there is a link between them, where probability $p(A_{i,j}) \approx 1$ indicates there exists an interaction with high confidence; conversely $p(A_{i,j}) \approx 0$ indicates a low interaction confidence. The prediction of a link between two given proteins can bet set up as a binary classification problem. The relations among nodes can be of several types; so, an edge of type r from node u to v can be defined as $u \xrightarrow{\text{r}} v \in \mathscr{E}$, which can be formulated as a multi-relational link prediction problem.

Using GNNs, one can map graph nodes into a low-dimensional vector space which may preserve both local graph structure and dissimilarities among node features. To address link prediction, one can employ a two layer encoder-decoder approach where the model learns Z from equation 25.5:

$$A' = DECODER(Z|P, A; \theta_{decoder}) \tag{25.6}$$

where $\theta_{decoder}$ denotes decoder (task)-specific learnable parameters, and $A'_{i,j}$ indicates the confidence score with the predicted link between protein i and j.

25.1.4.4 GNNs for Automated Function Prediction as a Multi-label Classification Problem

Given n-GO terms and m-proteins, $u = m - l$ proteins need to be annotated with term(s), whereas l proteins are already annotated. So for the i-th protein, the prediction will be $\mathrm{y}_i = \mathrm{y}_{i,1}, \mathrm{y}_{i,2}, ..., \mathrm{y}_{i,n}$ where $\mathrm{y}_{i,j} \in \{0,1\}$. This task can be considered as a binary multi-label classification problem, since a protein usually participates in multiple biological functions. This could be protein-centric, where GO-terms are annotated for each protein, or GO-term centric, where proteins are annotated for each GO-term, or protein-term pair centric, where a probability association score is predicted for each pair.

25.2 Highlighted Case Studies

In the following, we highlight three selected methods that exemplify SOTA techniques and performance.

25.2.1 Case Study 1: Prediction of Protein-Protein and Protein-Drug Interactions: The Link Prediction Problem

Liu et al (2019) apply a graph convolutional neural network (GCN) for PPI prediction as a supervised binary classification task. Learned representations of two proteins are fed to the model, and the model predicts the probability of interaction between the proteins. The model first captures position-specific information inside the PPI network and combines amino-acid sequence information to output final embeddings for each protein. The model encodes each amino acid as a one-hot vector and employs a graph convolutional layer to learn a hidden representation from the graph. To do that, Liu et al (2019) use the message passing protocol to update each protein embedding by aggregating the original features and first-hop neighbors' information, which is formulated as following:

$$X_1 = ReLU(D^{-1}\tilde{A}X_0W_0) \tag{25.7}$$

where $X_0 \in \mathbb{R}^{n \times n}$ is the original protein feature matrix which is an identity matrix; $X_1 \in \mathbb{R}^{n \times f}$ is the final output feature matrix, where f is the feature dimension of each protein after the graph convolution operation and W_0 is the trainable weight matrix. In the prediction phase, the authors utilize fully connected layers followed by batch normalization and dropout layers to extract high-level features; softmax is then used to predict the final interaction probability score. The experiments show that the method achieves mean AUPR (area under precision-recall curve) of 0.52 and 0.45 on yeast and human datasets, respectively, which outperforms sequence-based SOTA methods. Additionally, the authors report achieving 95% accuracy on yeast data under 93% sensitivity. Therefore, the extracted information from the PPI graph suggests that a single graph convolutional layer is capable of extracting useful information for the PPI prediction task.

Brockschmidt (Brockschmidt, 2020) proposes a novel GNN variant using feature-wise linear modulation (GNN-FiLM), originally introduced by Perez *et al.* (Perez et al, 2018) in the visual question-answering domain, and evaluates on three different tasks, including node-level classification of PPI networks. The targeted application in this work is the classification of proteins into known protein families or super-families, which is of great importance in numerous application domains, such as precision drug design. Typically, in GNN variants, the information is passed from the source to the target node considering the learned weights and the representation of the source node. However, the GNN-FiLM method proposes a hypernetwork, neural networks, that compute parameters for other networks (Ha et al, 2017), in graph settings, where the feature weights are learned dynamically based on the information that the target node holds. Therefore, considering function g as a learnable function to compute the parameters for the affine transformation, the update rule is defined for the l-th layer as follows:

$$\beta_{r,v}^{(l)} \gamma_{r,v}^{(l)} = g(h_v^{(l)}; \theta_{g,r}) \tag{25.8}$$

$$h_v^{(l+1)} = \sigma\left(\sum_{u \xrightarrow{\text{r}} v \in \mathscr{E}} \gamma_{r,v}^{(l)} \odot W_r h_u^{(l)} + \beta_{r,v}^{(l)}\right) \tag{25.9}$$

where g is implemented as a single linear layer in practice considering $\beta_{e,v}^{(t)}$ and $\gamma_{e,v}^{(t)}$ as the hyperparameters of the message passing operation in GNN, and $u \xrightarrow{\text{e}} v$ indicates that message is passing from u to v through a type r edge. In experiments, GNN-FiLM achieves micro-averaged F1 score of 99% which outperforms other variants when evaluated on protein classification tasks.

Zitnik et al (2018) employ GCNs to predict polypharmacy side effects, which emerge from drug-drug interactions when using drug combinations on patients' treatments. The problem can be formulated as a multi-relational link prediction problem in multimodal graph structured data. Specifically, Zitnik et al (2018) consider two types of nodes, proteins and drugs, and construct the network using protein-protein, protein-drug, and drug-drug interactions as polypharmacy side effects, whereas each side effect can be of different types of edges, called Decagon. More precisely, a relation of type r between two nodes (proteins or drugs), u and v, is defined as $(u, r, v) \in \mathscr{E}$. Here, the relations can be a side effect between two proteins, binding affinity of two proteins, or relation between a protein and a drug. More formally, given a drug pair (u, v), the task is to predict the likelihood of an edge, $A_{u,v} = (u, r, v)$. For this purpose, they develop a non-linear and multi-layer graph convolutional encoder to compute the embeddings of each node using original node features, called Decagon. To update a node's representation, authors transform the information of neighboring nodes by aggregation and propagation operations over the edges. The update operator is defined using the following rule:

$$h_i^{(l+1)} = \phi\left(\sum_r \sum_{j\in\mathcal{N}_r^i} c_r^{i,j} W_r^{(l)} h_j^{(l)} + c_r^i h_i^{(l)}\right) \tag{25.10}$$

where ϕ denotes non-linear activation function, $h_i^{(l)}$ indicates hidden state of the i-th node at the l-th layer, $W_r^{(l)}$ means relation-type specific learnable parameter matrix, $j \in \mathcal{N}_r^i$ are the neighboring nodes of i, $c_r^{i,j} = \frac{1}{\sqrt{|\mathcal{N}_r^i||\mathcal{N}_r^i|}}$ and $c_r^i = \frac{1}{\sqrt{|\mathcal{N}_r^i|}}$ are the normalization constant. Finally, a tensor factorization model is used to predict the polypharmacy side effects using these embeddings. The probability of a link of type r between node u and v is defined as:

$$\mathbf{x}_r^{u,v} = \sigma(g(u,r,v)) \tag{25.11}$$

where σ is the sigmoid function and g is defined as follows:

$$g(u,r,v) = \begin{cases} z_u^T D_r R D_r z_v & \text{if } u \text{ and } v \text{ both denote drug nodes} \\ z_u^T M_r z_v & \text{if any of } u \text{ or } v \text{ is not drug node} \end{cases} \tag{25.12}$$

where D_r, R and M_r are parameter matrices, such that D_r defines side-effect-specific diagonal matrix, R is global drug-drug interaction matrix, and M_r is relation-type-specific parameter matrix. Decagon achieves an AUPR of 83% under 80% precision, outperforming other baselines by up to 69%. The authors attribute the large margin in improvement to two components, the graph-structured convolution encoder and the tensor factorization model.

25.2.2 Case Study 2: Prediction of Protein Function and Functionally-important Residues

Automated Function Prediction (AFP) problems are often formulated as a multi-label classification problems and are more nuanced than predicting interactions between two proteins. Many works report that proteins connected in the same molecular network share the same functions (Schwikowski et al, 2000), but recent developments show that interacting proteins are not necessarily similar, and similar proteins do not necessarily interact (Kovács et al, 2019). Moreover, more than 80% of proteins interact with other molecules while functioning (Berggård et al, 2007). Therefore, identifying or predicting the roles of proteins in organisms is vital, and community-wide challenges have been organized to advance research towards this goal. These include the Critical Assessment of Function Annotation (CAFA) (Radivojac et al, 2013; Jiang et al, 2016; Zhou et al, 2019b) and MouseFunc (Peña-Castillo et al, 2008).

Many computation methods have been developed to this end to analyze protein-function relationships. Traditional machine learning approaches, such as SVMs (Guan et al, 2008; Wass et al, 2012; Cozzetto et al, 2016), heuristic-based methods (Schug, 2002), high dimensional statistical methods (Koo and Bonneau, 2018), and hierarchical supervised clustering methods (Das et al, 2015) have been extensively studied in AFP tasks and found that integration of several features, such as gene and protein network or structure outperforms sequence-based features. However, these traditional approaches rely strongly on hand-engineered features.

Deep learning methods have become prevalent. For example, DeepSite (Jiménez et al, 2017), Torng and Altman (2018), and Enzynet (Amidi et al, 2018) apply 3D convolutonal neural networks (CNNs) for feature extraction and prediction from protein structure data. However, storing the high-resolution 3D representation of protein structure and applying 3D convolutions over the representation is inefficient (Gligorijevic et al, 2020). Very recently, GCNs (Kipf and Welling, 2017b) (Henaff et al, 2015; Bronstein et al, 2017) have been shown to generalize convolutional operations on graph-like molecular representations and overcome these limitations.

In particular, Ioannidis et al (2019) adapt the graph residual neural network (GRNN) approach for a semi-supervised learning task over multi-relational PPI graphs to address AFP. The authors formulate a multi-relational connectivity graph as an $n \times n \times I$ tensor S, where $S_{n,n',i}$ captures the edge between proteins v_n and $v_{n'}$ for the i-th relation. The n proteins are encoded in a feature matrix $X \in \mathbb{R}^{n \times f}$, where the i-th protein is represented as an $f \times 1$ feature vector. Furthermore, a label matrix $Y \in \mathbb{R}^{n \times k}$ encodes the k labels. Subsets of proteins are associated with true labels, and the task is to predict the labels of proteins with unavailable labels. The neighborhood aggregation for the n-th protein and the i-th relation at the l-th layer is defined by the following formula:

$$H_{n,i}^{(l)} = \sum_{n' \in \mathcal{N}_n^{(i)}} S_{n,n',i} \check{Z}_{n',i}^{(l-1)} \tag{25.13}$$

where n' denotes the neighboring nodes of the n-th protein, and $\check{Z}_{n'i}^{(l-1)}$ denotes the feature vector of the n-th protein in the i-th relation at the l-th to the first layer. Neighboring nodes are defined as one-hop only, which essentially incorporates one-hop diffusion. However, successive operations eventually spread the information across the network. To apply multi-relational graphs, the authors combine $H_{ni}^{(l)}$ across i as follows:

$$G_{n,i}^{(l)} = \sum_{i'=1}^{I} R_{i,i'}^{(l)} H_{n,i'}^{(l)} \tag{25.14}$$

where $R_{i,i'}^{(l)}$ is the learnable parameter. Then, a linear operation mixes the extracted features as follows:

$$Z^{(l)} = G_{n,i}^T W_{n,i}^{(l)} - 1 \tag{25.15}$$

where $W_{n,i}$ is the learnable parameter. In summary, the neighborhood convolution and propagation step can be shown as:

$$Z^{(l)} = f(Z^{(l-1)}; \theta_z^{(l)}) \tag{25.16}$$

where $\theta_z^{(l)}$ is comprised of two weight matrices, W and R, which linearly combine the information of neighboring nodes and the multi-relational information, respectively. Moreover, the authors incorporate residual connection to diffuse the input, X, across L-hop neighborhoods to capture multi-type diffusion; that is:

$$Z^{(l)} = f(Z^{(l-1)}; \theta_z^{(l)}) + f(X; \theta_x^{(l)}) \tag{25.17}$$

A softmax classification layer is used for the final prediction. The authors apply this model on three multi-relational networks, comprising generic, brain, and circulation cells. The model is shown to perform better than general graph convolutional neural networks.

Recently, Gligorijevic et al (2020) employ DeepFRI, based on GCNs, for functionally annotating protein sequences and structures. DeepFRI outputs probabilities for each function. A Long Short-Term Memory Language Model (LSTM-LM) (Graves, 2013) is pretrained on around 10 million protein sequences from protein family database (Pfam) (Finn et al, 2013) to extract residue-level position-context features. The following equation is used:

$$H^0 = H^{input} = ReLU(H^{LM} W^{LM} + XW^X + b) \tag{25.18}$$

where H^0 is the final residue-level feature representation and the first graph convolutional layer. W^{LM}, W^X and b are learnable parameters trained with the graph convolutional layers. Contact-map features, which encode tertiary protein structure, combined with LSTM-LM task-agnostic sequence-embeddings are fed to a GCN while keeping LSTM-LM frozen. The l-th layer of the convolution takes sequence-embeddings and the contact map A and outputs residue-level embeddings to the next, $(l+1)$-th, layer. Residue level features are extracted by propagating residue information to proximal residues. The rule for updating the node representation is:

$$H^{(l+1)} = ReLU(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}) \tag{25.19}$$

The features are then concatenated into a single feature matrix as a protein embedding. Intuitively, embeddings from different layers can be thought as context-aware features. Additionally, the feature extraction strategy exploits linear or non-linear relationships from neighbouring residues, as well as residues distant in sequence but proximal in structure.

The learned protein representation is fed into two consecutive fully connected layers to obtain predictions as class probabilities for all the GO-terms. The authors evaluate their model on experimental and predicted structures and compare with existing baseline models, including CAFA-like BLAST (Wass et al, 2012) and CNN-based sequence-only DeepGOPlus (Kulmanov and Hoehndorf, 2019), on each sub-ontology of GO-terms and EC numbers and outperform in every category.

Zhou et al (2020b) apply a GCN model, DeepGOA, to predict maize protein functions. The authors exploit both GO structure information and protein sequence information for a multi-label classification task. Since GO organizes the functional annotation terms into a directed acyclic graph (DAG), the authors utilize the knowledge encoded in the GO hierarchy. First, amino acids of a protein are encoded into one-hot encodings, a 21-dimensional feature vector for each amino acid, as there are 20 amino acids and sometimes there are undetermined amino acids in a protein. Proteins might be different in length; therefore, the authors only extract the first 2000 amino acids for those proteins which are longer than that. Otherwise, the encodings are zero-padded. So the *i*-th protein is represented as

$$X_i = [x_{i1}, x_{i2}, x_{i3}, ...x_{i2000}] \tag{25.20}$$

To learn the low-dimensional feature representation of each protein sequence, the authors apply CNNs of four different sizes of convolutional kernels, such as 8, 16, 24 and 32, to extract hypothetical non-linear secondary or tertiary structure information. The 1D convolution operation is formulated as follows:

$$c_{im} = f(w * x_{i(m:m+h)}), m \in [1, k-h] \tag{25.21}$$

where h is the sliding window length, $w \in \mathbb{R}^{21\times h}$ is a convolutional kernel, and $f(\cdot)$ is a non-linear activation function. Then, the authors incorporate the GO structure into the model. To do that, graph convolutional layers are deployed to generate the embeddings of the GO terms by propagating information among GO terms using neighboring terms in the GO hierarchy. For τ number of GO terms, initial one-hot feature description, $H^0 \in \mathbb{R}^{\tau\times\tau}$, and correlation matrix, $A \in \mathbb{R}^{\tau\times\tau}$ are computed as input. For the l-th layer's representation, H^l is updated using the following neighborhood information propagating equation:

$$H^l = f(\hat{A}H^{l-1}W^l) \tag{25.22}$$

where $\hat{A} \in \mathbb{R}^{\tau\times\tau}$ is the symmetrically normalized correlation matrix derived from A, $f(\cdot)$ is a non-linear activation function, and $W^l \in \mathbb{R}^{d_{l-1}\times d_l}$ is the learnable transformation matrix. Then, such graph convolutional layers are stacked to capture high- and low-order information of the GO DAG. In this way, DeepGOA learns a semantic representation of GO-terms, $H \in \mathbb{R}^{\tau\times d}$, and protein sequence representation, $Z \in \mathbb{R}^{n\times d}$, in some d-dimensional semantic space. Dot product is used to then compute protein-term pair association probabilities as follows:

$$\hat{Y} = HZ^T \tag{25.23}$$

Cross-entropy loss for the multi-label loss function is used to train the model end-to-end. The authors experiment on the Maize PH207 inbred line (Hirsch et al,

2016) and the human protein sequence dataset and show that DeepGOA outperforms SOTA methods.

25.2.3 *Case Study 3: From Representation Learning to Multirelational Link Prediction in Biological Networks with Graph Autoencoders*

Yang et al (2020a) employ signed variational graph auto-encoder (S-VGAE) to automatically learn graph representation, and incorporate protein sequence information as features for the PPI prediction task. The authors report SOTA performance compared to existing sequence-based models on several datasets.

The protein interaction network is encoded as an undirected graph, with different signs (i.e., positive, negative or neutral) added the edges in the adjacency matrix to extract fine-grained features, where the model is assumed to learn negative impact of highly negative interactions. Moreover, the authors consider only high-confidence interactions in the cost function, enabling the model to learn embeddings more accurately. First, protein sequences are encoded using the CT method (Shen et al, 2007). All amino acids are divided into seven categories considering their dipole and side-chain volumes. Each group represents analogous mutations due similar characteristics. Thus, a protein can be represented as a sequence of numbers representing a category. Then, a window of size 3 amino acids slides over the numeric sequence one step at a time and counts the number of occurrences of each triad. Thus, the size of a protein CT vector is 343(=m), which can be defined as follows:

$$V = [r_1, r_2, ...r_M] \tag{25.24}$$

where r_i is the number of occurrences of each triad type. For n proteins, the input features of each protein can be summarized in a matrix $X \in \mathbb{R}^{n\times m}$. Afterwards, S-VGAE is employed to extract protein embeddings by combining both graph structure and sequence information, following Kipf and Welling's (Kipf and Welling, 2016) variational graph auto-encoder. Considering the primary/sequence features, its neighborhood structures and positions in the graph, the encoder maps each protein x_i to a low-dimensional vector z_i. The idea is to map proteins' original features X into low dimensional embeddings Z using an augmented information adjacency matrix A. The encoding rule is formulated as follows:

$$q(Z|X,A) = \prod_{i=1}^{N} q(z_i|Z,A) \tag{25.25}$$

$$q(z_i|Z,A) = \mathcal{N}(z_i|\mu_i, diag(\sigma_i^2)) \tag{25.26}$$

Mean vector, μ_i, and standard deviation vector, σ_i, is defined as follows:

$$\mu = GCN_\mu(X,A) \tag{25.27}$$

$$\log \sigma = GCN_{\sigma}(X,A) \tag{25.28}$$

where *GCN* is a neighborhood aggregation propagation step formulated as below:

$$GCN(X,A) = AReLU(AXW_0) \tag{25.29}$$

$$GCN_{\mu}(X,A) = AReLU(AXW_1) \tag{25.30}$$

$$GCN_{\sigma}(X,A) = AReLU(AXW_2) \tag{25.31}$$

where W_0, W_1 and W_2 are trainable parameters and, GCN_{μ} and GCN_{σ} share W_0 to reduce parameters. The decoder predicts the classification label of protein i and j by taking the dot product of their lower-dimensional embeddings z_i and z_j; the interaction probability indicates whether there is a connection between two proteins. This is defined as follows:

$$p(A|Z) = \prod_{i=1}^{N}\prod_{j=1}^{N} p(A_{i,j}|z_i, z_j) \tag{25.32}$$

$$p(A_{i,j} = 1|z_i, z_j) = \sigma(z_i^T z_j) \tag{25.33}$$

where $\sigma(\cdot)$ is the logistic sigmoid function. Thus, the S-VGE learns to encode protein embeddings into low-dimensional features by solving the task of decoding the learned embeddings back to the original graph structure. Instead of using the decoder as the final classification layer, the authors utilize it as a generative model for learning latent features. Then, three fully connected layers perform the final classification task. Overall, the model achieves more than 98% accuracy on five different datasets.

Hasibi and Michoel (2020) propose a graph feature auto-encoder (GFAE) model, called FeatGraphConv, which is trained on a feature reconstruction task instead of graph reconstruction task. The model performs well on predicting unobserved node features on biological networks, such as transcriptional, protein-protein and genetic interaction networks. FeatGraphConv investigates how well GNNs might preserve node features. The authors aim to identify whether or not the graph structure and feature values encode similar information. The relationship between a graph G and latent embeddings Z can be formulated using graph convolutional layers as messaging passing protocol by aggregating neighborhood information as follows:

$$Z = GCN(G;\theta) = GCN(X,\tilde{A};\theta) \tag{25.34}$$

$$Z = \sigma(\tilde{A}ReLU(\tilde{A}XW_0)W_1) \tag{25.35}$$

where θ contains learnable weights, defined as $\theta = W_0; W_1; ......W_i$, and σ is a non-linear task-specific mapping function. The authors leverage four message passing and neighborhood information aggregation operations. The GCN update rule (Gilmer et al, 2017) is followed for the i-th protein's representation, h_i^k, at the l-th layer as follows:

$$h_i^l = \sum_{j \in \mathcal{N}(i) \cup i} \frac{1}{\sqrt{deg(i)} * \sqrt{deg(j)}} W h_j^{l-1} \tag{25.36}$$

The GraphSAGE (Hamilton et al, 2017b) update rule is then deployed:

$$h_i^l = W_1 h_i^{l-1} + W_2 Mean_{j \in \mathcal{N}(i) \cup i} h_j^{l-1} \tag{25.37}$$

Additionally, the authors employ the GraphConv (Morris et al, 2020b) operator:

$$h_i^l = W_1 h_i^{l-1} + \sum_{j \in \mathcal{N}(i)} W_2 h_j^{l-1} \tag{25.38}$$

A new update rule is also proposed:

$$h_i^l = W_2 (W_1 h_i^{l-1} || Mean_{j \in \mathcal{N}(i) \cup i} (W_1 h_j^{l-1})) \tag{25.39}$$

where $(\cdot||\cdot)$ denotes a concatenation operation. The authors train the learnable parameters on the embeddings ability to reconstruct the adjacency matrix, which is formulated as follows:

$$\hat{A} = Sigmoid(ZZ^T) \tag{25.40}$$

Cross-entropy loss between A and $\hat{A}$ and gradient descent are used to update the weights. Finally, the embeddings Z are used to predict the class Y in predicting missing links in the adjacency matrix and thus in the graph.

25.3 Future Directions

As this survey indicates, many variants of GNNs have been applied to obtain information on protein function. Much work remains to be done. Future directions can be broadly divided into two categories, methodology-oriented and task-oriented.

Many existing GNN-based approaches are limited to proteins of the same size (number of amino acids). This essentially weakens model capacity for the particular task at hand. Therefore, future research needs to focus on size-agnostic, as well as task-agnostic models. Choosing the right model is always a difficult task. However, benchmark datasets and available packages are making it easier to develop models expediently.

Enhancing model explainability is also an important direction. Some community bias has been observed towards focusing model development on GCNs for learning semantic and topological information for the function prediction task. However, there are many other variants of GNNs. For instance, graph attention networks may prove useful. Existing literature also often ignores ablation studies, which are important to provide a strong rationale for choosing a particular component of the model over others.

Most of the PPI prediction tasks assume training a single model for an organism. Leveraging multi-organisms PPI networks provides more data and may result in better performance. In the same spirit, leveraging multi-omics data combined with sequence and structural data may advance the state of the art.

Finally, we draw attention to the site-specific function prediction task, which provides more information and highlights specific residues that are important for a particular function. This fine-grained function prediction task can be even more critical to support other tasks, such as drug design. Transfer learning across related tasks may additionally provide insights for learning important attributes.

This work is supported in part by National Science Foundation Grant No. 1907805 and Grant No. 1763233. This material is additionally based upon work by AS supported by (while serving at) the National Science Foundation. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

Editor's Notes: In addition to small molecules introduced in Chapter 25, large molecules such as proteins and DNA represent another domain in bioinformatics that started to largely leverage the techniques from graph neural networks. The recent popularity of graph deep learning for small and large molecules seems to share similar reasons. The first reason is the well-formulated problem and the availability of benchmark datasets while the other is due to the high complexity of the problem and the insufficiency of existing techniques. On the other hand, there is also some subtle difference between them: The deep graph learning community seems dedicated to more extensive new models for small molecules than large ones previously. But in recent years, research frontiers tend to start to transfer the success in small molecules to benefit larger ones, with representative works such as AlphaFold.