


## TECHNICAL NOTE

# Graph2GO: a multi-modal attributed network embedding method for inferring protein functions

Kunjie Fan<sup>1</sup>, Yuanfang Guan,<sup>2,\*</sup> and Yan Zhang<sup>1,3,\*</sup>

<sup>1</sup>Department of Biomedical Informatics, College of Medicine, The Ohio State University, Columbus, OH 43210, USA; <sup>2</sup>Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI 48109, USA and <sup>3</sup>The Ohio State University Comprehensive Cancer Center (OSUCCC - James), Columbus, OH 43210, USA

\*Correspondence address. Yuanfang Guan, Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI 48109, USA. E-mail: [gyuanfang@umich.edu](mailto:gyuanfang@umich.edu); Yan Zhang, Department of Biomedical Informatics, College of Medicine, The Ohio State University, Columbus, OH 43210, USA. E-mail: [yanzhang.biomed@gmail.com](mailto:yanzhang.biomed@gmail.com)  <http://orcid.org/0000-0002-3357-5121>

## Abstract

**Background:** Identifying protein functions is important for many biological applications. Since experimental functional characterization of proteins is time-consuming and costly, accurate and efficient computational methods for predicting protein functions are in great demand for generating the testable hypotheses guiding large-scale experiments. **Results:** Here, we propose Graph2GO, a multi-modal graph-based representation learning model that can integrate heterogeneous information, including multiple types of interaction networks (sequence similarity network and protein-protein interaction network) and protein features (amino acid sequence, subcellular location, and protein domains) to predict protein functions on gene ontology. Comparing Graph2GO to BLAST, as a baseline model, and to two popular protein function prediction methods (Mashup and deepNF), we demonstrated that our model can achieve state-of-the-art performance. We show the robustness of our model by testing on multiple species. We also provide a web server supporting function query and downstream analysis on-the-fly. **Conclusions:** Graph2GO is the first model that has utilized attributed network representation learning methods to model both interaction networks and protein features for predicting protein functions, and achieved promising performance. Our model can be easily extended to include more protein features to further improve the performance. Besides, Graph2GO is also applicable to other application scenarios involving biological networks, and the learned latent representations can be used as feature inputs for machine learning tasks in various downstream analyses.

**Keywords:** protein function prediction; graph neural network; attributed network embedding; representation learning; multi-modal model

## Introduction

Knowledge of protein functions is of great importance to understanding life at a molecular level, studying disease mechanisms, and helping explore novel therapeutic targets. However, the experimental identification of protein functions is time-consuming and expensive, which is not suitable for large-scale applications. Therefore, high-throughput computational methods are required for discovering protein functions with reason-

able quality and accuracy [1], and providing testable hypotheses for targeted experimental validation.

Most existing computational algorithms exploit homology inference to infer protein functions [2–4], which are based on the assumption that proteins with similar sequences frequently carry out similar functions. A standard approach is simply to transfer annotations from the best-annotated BLAST hit [5]. Some approaches involve the use of protein-protein interaction

Received: 14 December 2019; Revised: 30 April 2020

© The Author(s) 2020. Published by Oxford University Press. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.

(PPI) networks, based on the fact that proteins closer in the network have a greater chance of sharing similar functions [6, 7]. Some other methods utilize domain content in the sequence to assign functions [8, 9]. Under the assumption that domains are protein structural architecture modules, it makes sense that protein function should be closely related to or determined by the composite domains. There are also other methods making use of other sources of information for protein function prediction, such as protein subcellular localization [10, 11] and post-translational modifications [11], as well as the literature [12].

Given the limited prediction capacity of a single source of information, many methods have been proposed to combine several kinds of information and take advantage of the power of machine learning techniques. For instance, INGA [13] performs sequence similarity and domain architecture searches, and combines them with enrichment analyses on interaction networks to derive the consensus prediction. Similarly, COFACTOR [14] consists of three individual pipelines for sequence-, structure-, and PPI-based predictions by querying the UniProt-GOA [15], BioLip, and STRING [16] databases, respectively, and it generates the consensus based on three confidence scores obtained from the three pipelines. DeepGO [17] uses representation learning methods to learn features from sequence and interaction networks respectively, and then combines them to predict the function using a deep neuro-symbolic model. Although these methods could achieve reasonable prediction accuracy, a limitation is that they only treat multiple kinds of features separately and do not consider the relationships between proteins for each feature, which might result in the loss of information contained within the interactions. A possible solution to take into account the relationships is to use PPI networks to transfer features between proteins.

Graphs, such as those representing social networks, molecular graph structures [18], and biological PPI networks, occur naturally in various real-world scenarios and have important applications in modern machine learning. Modeling the interactions between entities as graphs has enabled researchers to understand the network system in a systematic manner. For example, in a social network application one might wish to predict the role of a person or recommend new friends to a user [19]; in a clinical application, researchers want to predict new therapeutic applications of drug molecules [20]; and basic scientists are also interested in classifying the roles of a protein in a biological interaction graph [21]. Mashup [22] is a framework for scalable and robust multiple network integration, and generates low-dimensional vectors for nodes by characterizing topological contexts in heterogeneous PPI networks based on the random walk with restart method. The key of Mashup is the use of the random walk with restart method to analyze the network structure. Researchers also proposed novel methods for dimension reduction and the integration of heterogeneous networks by solving optimization problems. DeepNF [23] is a network fusion method for extracting protein features from multiple heterogeneous interaction networks based on multi-modal deep auto-encoders. DeepNF also utilizes the random walk with restart to extract information about network structures for individual PPI networks. DeepNF proposed a different integration method based on auto-encoders. Both of these two methods adopt a two-stage model: first generating informative embeddings based on network structures in an unsupervised manner, then building a supervised classification model to predict gene ontology (GO) terms with embeddings as the input. Although both Mashup and DeepNF can generate embeddings used for downstream protein function prediction, they only consider the topological informa-

tion contained in multiple PPI networks, while ignoring informative protein attributes, such as protein sequence or protein domains; thus, they might lack important information.

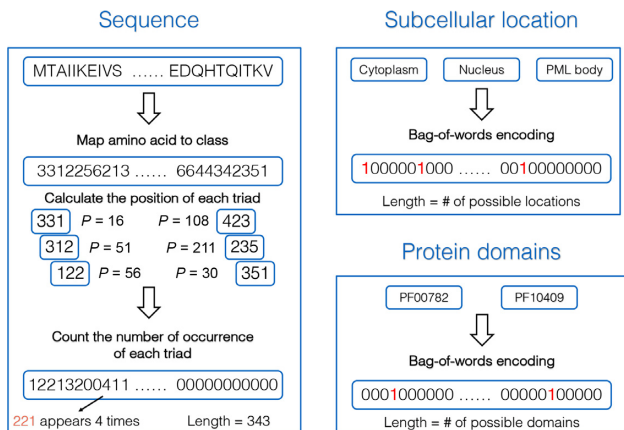
A crucial challenge in machine learning on graphs is finding a way to incorporate multiple types of information about the structure and attributes of the graph into the machine learning model. Traditional approaches usually use summary graph statistics [24], kernel functions [25], or handcrafted features to represent local neighborhood structures. Nowadays, people seek to learn representations that encode structural information in a data-driven way. The idea behind these representation learning methods is to learn a function that maps nodes in the graph to points in a low-dimensional vector space. By optimizing this mapping, the geometric relationships in the learned space can reflect the original structure of the graph, and the learned representation can be used as feature inputs for downstream machine learning applications [26]. These network representation learning methods can also make use of node attributes to generate more informative embeddings, such as user profiles in a social network and protein signatures in a protein interaction network [21].

In this paper, we propose Graph2GO, a multi-modal graph-based architecture, that can make use of several kinds of data sources in a unified way to predict protein function. Unlike other consensus methods that we mentioned earlier, we first use PPI and sequence similarities to build two graphs separately and use protein sequence information, protein subcellular location, protein domains, or any other possible information as node attributes in both graphs. Given the attributed graph, we use the attributed network representation learning algorithm to obtain informative embeddings for each node in each graph. In the graph, a node represents a protein. In this way, we manage to learn representations by modeling both node attributes and network structures comprehensively and simultaneously. Then, we combine these two embeddings and use them to predict the protein functions with a feedforward neural network model.

As far as we know, we are the first to use attributed network representation learning methods to model both an interaction network and a sequence similarity network (SSN) with node attributes and predict protein functions, and we have successfully achieved state-of-the-art performance on the benchmark data set. Besides, our model is extensible to other function-related information to further improve the performance. Graph2GO does not rely on any manually crafted features and is entirely data driven. Graph2GO is also applicable to other similar scenarios, such as predicting new therapeutic applications of existing drugs, since the learned embeddings can be used as feature inputs for various downstream machine learning tasks.

## Materials and Methods

Graph2GO consists of two parts: the first part is an unsupervised graph-based representation model that utilizes both network information (PPI, sequence similarities) and node attributes (protein sequence, subcellular location, and protein domains) to generate unique embedding vectors for each protein; the second part is a fully-connected deep neural network (DNN) classifier, which use embeddings as features and gene ontology (GO) [27] as function labels. GO defines concepts that describe functions and classifies functions on three aspects: molecular function (MF), cellular component (CC), and biological process (BP) [27]. We first describe how we obtain and encode different kinds



**Figure 1:** Here we use PTEN as an example to show how we encode sequence, subcellular location, and protein domains features. For sequence encoding, we use the conjoint triad (CT) method. For subcellular location and protein domains, we use bag-of-words encoding.

of information, and then describe the detailed model specification. In Fig. 1, we use the PTEN gene as an example to detail how we encode these information. The model architecture is shown in Fig. 2.

## Data set

We use reviewed and manually annotated proteins from SwissProt (release 2018.11) [15]. The data set contains 20,412 human proteins. The data set provides GO annotations along with the experimental evidence code. We also obtained protein sequences, subcellular locations, and protein domains (Pfam) from SwissProt. We downloaded PPI networks from the STRING database (v10.5) [16], filtered by the proteins we obtained from SwissProt.

## Data encoding

### Protein-protein interaction network

From the network data in STRING, we use the “combined score” provided by STRING as the confidence score. As shown in Supplementary Figure S2, We perform cross-validation to choose the threshold of the combined score for building the PPI network, and we only use interactions that have a “combined score” greater than 300 to construct the adjacency matrix as the representation of the network. For interactions with low confidence and protein pairs with no interactions mentioned in STRING, we assign the corresponding element in the adjacency matrix as 0.

### Sequence similarity network

In order to build a SSN, we use the BLAST program to search similar sequences for each protein in our data set [28]. We only select similarity pairs that have an “e-value” smaller than  $1e-4$  as candidate edges for the similarity network. Detailed cross-validation results on selecting the threshold are shown in Supplementary Figure S1.

### Protein sequence

We encode amino acid sequences following the conjoint triad (CT) method [29], which has been widely used to represent sequences in related fields [30–33]. The 20 kinds of amino acids are first clustered into 7 classes according to the dipoles and volumes of the side chains, since amino acids within the same class

likely involve synonymous mutations. And then all the amino acids in the same class are considered as identical. The mapping between classes and amino acids is shown in Supplementary Table S1. Then, we consider any 3 continuous amino acids as a unit and count the triad frequencies by calculating the occurrence numbers within the protein sequence. Thus, the dimension of the CT encoding is  $7 \times 7 \times 7 = 343$ . Using the CT method, we manage to convert amino acid sequences into fixed-dimension representation.

### Subcellular location

We obtained subcellular location information from the SwissProt data set. In total, there are 359 different subcellular locations. We use bag-of-words encoding to represent this information; that’s to say, the location is encoded as a binary vector of length 359 with each element indicating whether the protein is annotated with this location. Therefore, for a protein without any subcellular location annotation, it is represented as a vector of all 0s.

### Protein domains

In the data set we obtained from SwissProt, there are 5,817 unique protein domains. In order to avoid the curse of dimensionality and to decrease the complexity, we only use protein domain terms that appear more than 5 times in our data set, and we are left with 655 terms. The protein domain knowledge is also encoded by way of bag-of-words encoding.

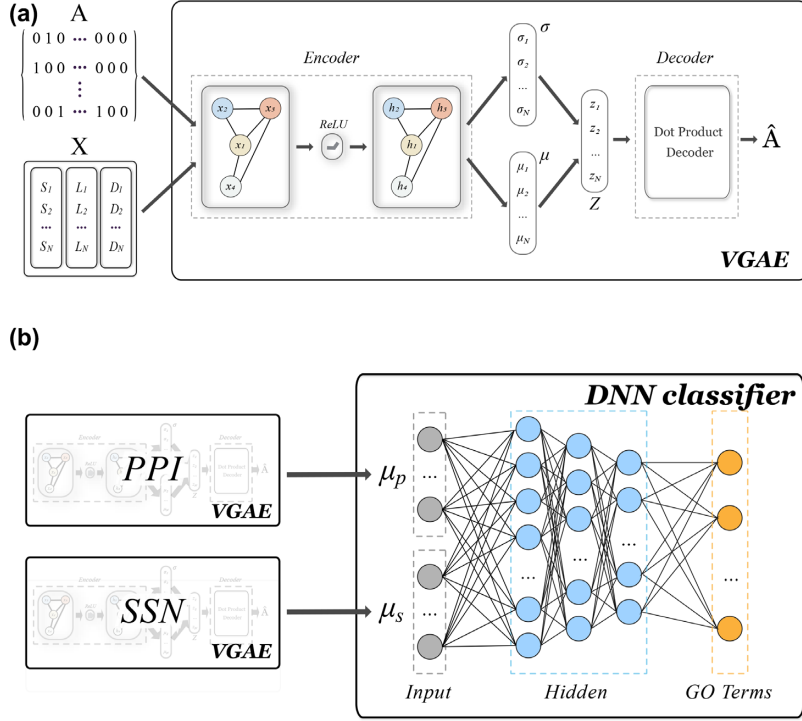
## VGAE model

In the first part of Graph2GO, the core is a variational graph auto-encoder (VGAE) [34], which can generate latent representations based on both network structure and node features, as shown in Fig. 2a. We will discuss this model by problem formulation, followed by its inference part (encoder) and generative part (decoder). The purpose of VGAE is to learn interpretable embedding for each protein by training the encoder and decoder at the same time.

### Problem formulation

We are given an undirected graph,  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , with  $N = |\mathcal{V}|$  nodes. Here,  $N$  is the number of proteins, and each vertex of  $\mathcal{G}$  represents one protein, while each edge is one interaction in the PPI network or a similarity pair in the SSN. The adjacency matrix  $\mathbf{A}$  of  $\mathcal{G}$  and its degree matrix  $\mathbf{D}_A$  are derived from known network information. We enforce self-loops in the graph by simply adding the identity matrix to  $\mathbf{A}$ . The input features of each vertex are included in an  $N \times R$  matrix  $\mathbf{X}$  which is the concatenation of protein sequence feature  $\mathbf{S}$ , subcellular location feature  $\mathbf{L}$ , and protein domains feature  $\mathbf{D}$ , as shown in Fig. 2a. Here,  $R$  is the sum of feature dimensions of matrix  $\mathbf{S}$ ,  $\mathbf{L}$ , and  $\mathbf{D}$ .

The model depicted in Fig. 2a is basically an encoder-decoder model. First the encoder maps each node  $v_i$  in the graph to a low-dimensional latent variable  $z_i$ , based on the node’s position in the graph, its local neighborhood structure, and its attributes. Next, the decoder reconstructs the adjacency matrix term  $A_{ij}$  corresponding to the pair of nodes  $v_i$  and  $v_j$ . By jointly optimizing the encoder and decoder, the model learns to compress graph structure information and original node attributes into the low-dimensional latent space. In principle, the model learns to propagate features across the proteins based on the network structure, and the encoder-decoder architecture ensures that the learned representations in the latent space are meaningful and informative.



**Figure 2:** (a) Architecture of the variational graph auto encoder (VGAE), the first part of Graph2GO. The inputs to VGAE include an adjacency matrix  $A$ , representing a protein-related network, and a node attribute matrix  $X$ . VGAE is an encoder-decoder approach. The encoder is a two-layer graph convolutional network and the decoder is a dot product decoder. The mean vector,  $\mu_i$ , is the output embedding for classification. (b) Graph2GO pipeline consisting of two VGAE models for PPI networks and SSN respectively, and the final DNN classifier. The two embeddings generated from the PPI network and SSN are concatenated as the input for the DNN classifier, and the DNN classifier outputs the probabilities of the protein having each GO term annotation.

### Encoder

The inference module is a graph convolutional network (GCN) encoder [35], which is a function with the goal of a mapping from the original features  $X$  to the latent variable  $Z$  with the network information  $A$ . To be more specific, we want to learn a probability model  $q(Z|X, A)$ . Here we use GCN  $g$  to model this probability:

$$[\mu; \log \sigma] = g(X, A; \phi) \quad (1)$$

$$q(Z|X, A) = \mathcal{N}(Z; \mu, \sigma^2 I) \quad (2)$$

where  $q$  is a function that encodes proteins into latent variables  $Z$  based on network information  $A$  and node attributes  $X$ ,  $\phi$  is the parameter of GCN  $g$ , and  $I$  is an identity matrix.  $\mu$  and  $\sigma$  are the mean and variance, respectively, of the Gaussian distribution corresponding to latent variable  $Z$ , and are estimated using network  $g$  from data directly. Then  $Z$  can be sampled from  $q(Z|X, A)$ . According to the reparameterization trick [36],  $z_i$  is obtained by:

$$z_i = \mu + \sigma \odot \epsilon_i \quad (3)$$

where  $\odot$  is element-wise multiplication and  $\epsilon_i \sim \mathcal{N}(0, I)$ .

Our GCN  $g$  is a two-layer network as defined:

$$g(X, A; \phi) = \tilde{A} \text{ReLU}(\tilde{A}XW^{(0)})W^{(1)} \quad (4)$$

where  $W^{(i)}$  are parameter matrices we need to train,  $\text{ReLU}(\cdot) = \max(0, \cdot)$  is the activation function, and  $\tilde{A} = D_A^{-\frac{1}{2}} A D_A^{-\frac{1}{2}}$  is the symmetrically normalized adjacency matrix [35].

The intuition of GCN is as follows. The multiplication of feature matrix  $X$  and the adjacency matrix  $A$  means that, for every node, we sum up feature vectors of all its neighboring nodes and

itself to update its feature representation. In this way, node attributes are propagated across the graph and the information associated with each protein is augmented, especially for proteins with little annotation. This graph convolution operation is similar to Laplacian smoothing, which makes features of nodes in the same cluster similar [37]. In order to avoid changing the scale of the feature vectors,  $A$  is first normalized so that all rows sum to 1 by symmetric normalization. Multiplying  $X$  with  $\tilde{A}$  means taking the average of neighboring nodes' features. In this way, GCN can effectively learn embeddings through integrating neighboring graph features.

### Decoder

As our latent embedding already contains both node attributes and structure information, the generative module we define here is a simple inner product decoder that aims to reconstruct adjacency matrix  $A$  using learned latent variables  $z_i$ :

$$p(A|Z) = \prod_{i=1}^N \prod_{j=1}^N p(A_{ij}|z_i, z_j) \quad (5)$$

$$p(A_{ij} = 1|z_i, z_j) = \sigma(z_i^\top z_j) \quad (6)$$

where  $\sigma(\cdot)$  is the logistic function. We use the logistic function-transformed inner product of  $z_i$  and  $z_j$ , shown on the right-hand side of Equation (6), as the probability of these two proteins having interaction. As indicated in Fig. 2a, the output of the decoder  $\hat{A}$  is the approximation of adjacency matrix  $A$ , and we optimize the model so as to make them as close as possible.

### Cost function

Similar to variational auto-encoder (VAE) [36], the cost function is the reconstruction error with a regularizer:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}[\log p(\mathbf{A}|\mathbf{Z})] - \text{KL}[q(\mathbf{Z}|\mathbf{X},\mathbf{A})\|p(\mathbf{Z})] \quad (7)$$

where  $\text{KL}[q(\cdot)\|p(\cdot)]$  is the Kullback-Leibler divergence between  $q(\cdot)$  and  $p(\cdot)$ . The first term is to minimize the reconstruction error of the adjacency matrix  $\mathbf{A}$ . The second term is to minimize the difference between  $q(\mathbf{Z}|\mathbf{X},\mathbf{A})$  and  $p(\mathbf{Z})$ . The cost function is the tradeoff between how accurately our model can reconstruct the input network and how closely the latent variables can match  $p(\mathbf{Z})$ . As specified in VAE, we assume  $p(\mathbf{Z}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ . We train the VGAE using stochastic gradient descent to optimize the cost function with respect to the parameters of the encoder [36].

### DNN classifier

In the second part of Graph2GO, as shown in Fig. 2b, we take out embeddings  $\mu_i$  contained in the matrix  $\mu$  and train a fully-connected DNN as the final supervised classifier. DNN has become one of the most popular and powerful techniques for supervised machine learning [38]. It consists of three types of layers: an input layer, hidden layers, and an output layer. The inputs to the classifier are embedding vectors for each protein, while the output layer represents GO terms that we aim to predict. Hidden layers are stacked between the input layer and output layer, aiming to learn meaningful abstract features for the task. To go from one layer to the next, a set of units (neurons) compute a weighted sum of their inputs from the previous layer and pass the result through a non-linear function (activation function). By stacking multiple hidden layers, DNN can learn an extremely intricate non-linear function mapping from the input to the output, which means it works best when the task is inherently non-linear. Gradient descent algorithm is commonly used for training neural networks and binary cross-entropy is used as the cost function in our multi-label classification task.

We train three classifiers: one each for MF, BP, and CC respectively. For each classifier, it is a multi-class, multi-label model, and the dimension of the output space is the number of GO terms within each ontology. Each protein may be predicted with multiple GO terms simultaneously. The classifier predicts the probabilities of the protein having each GO term annotation. The performance of the classifier can be excellent without many complex neural network structures, since the embeddings already contain enough information and are highly representative in the learned low-dimensional vector space.

### Graph2GO pipeline

As shown in Fig. 2b, the Graph2GO pipeline consists of two VGAE models for the PPI network and SSN, and the final DNN classifier for predicting protein functions. Instead of combining two networks first and training one VGAE to obtain overall embeddings, by training an independent VGAE model for each network and combining their embeddings, we try to avoid introducing noise and to keep as much information as possible. We compared the performance of these two ways for integrating two networks and the detailed results are shown in Supplementary Table S2.

## Results

### Experimental setup

Graph2GO was implemented using Tensorflow in Python and took advantage of the powerful computing capacity of a graph-

ics processing unit (GPU). All the simulations were carried out on the Owens cluster provided by the Ohio Supercomputer Center [1] with 27 processors and 127 GB memory. The GPU we used was a NVIDIA Tesla P100 with 16 GB memory. Our source code is available at <https://github.com/yanzhanglab/Graph2GO>.

For our experiments, we use SwissProt's reviewed and manually annotated human proteins (release 2018.11). We only use proteins that also exist in the STRING database (v10.5), where corresponding interaction information is available. In order to use the CT method to encode amino acid sequences, we delete proteins whose sequence contains ambiguous amino acids, including B, O, J, U, X, and Z. After filtering out 5,279 proteins, our final data set contains 15,133 proteins, along with 1,713,652 PPIs and 843,212 edges in the SSN. For GO annotations, we only consider the experimental evidence code among EXP, IDA, IPI, IMP, IGI, IEP, TAS, and IC [27]. If a protein is annotated with a GO term, we additionally annotated it with all the ancestor terms.

In the first part of our architecture, the encoder is a two-layer neural network structure with one layer having 400 neurons and the other having 800 neurons. Using two layers of graph convolutional operations is recommended in Kipf and Welling [35] and Li et al. [37], and we confirm the choice of two layers by performing cross-validation, as shown in Supplementary Figure S1. We initialize weights as described in Glorot and Bengio [39], and train the model for 200 iterations using Adam algorithm with a learning rate of 0.001 [40]. The final prediction classifier is a three-layer fully connected neural network with each layer having 1,024, 512, and 256 neurons respectively. A dropout layer and a batch normalization layer are used between every set of dense layers to avoid over-fitting and the dropout rate is set as 0.3. Adam is used to train the model for 100 iterations with a learning rate of 0.001.

### Evaluation metrics

We use metrics that are similar to those adopted by the Critical Assessment of protein Function Annotation algorithms (CAFA) challenge to evaluate our model and compare it with others [1]. The first metric we use is the maximum F-measure (F-max) over all possible thresholds, defined as follows:

$$pr(t) = \frac{\sum_i \sum_f I(f \in P_i(t) \wedge f \in T_i)}{\sum_i \sum_f I(f \in P_i(t))} \quad (8)$$

$$rc(t) = \frac{\sum_i \sum_f I(f \in P_i(t) \wedge f \in T_i)}{\sum_i \sum_f I(f \in T_i)} \quad (9)$$

$$F_{\max} = \max_t \left\{ \frac{2 \cdot pr(t) \cdot rc(t)}{pr(t) + rc(t)} \right\} \quad (10)$$

where  $pr$  means precision,  $rc$  means recall,  $I$  is the indicator function,  $f$  is a GO term,  $P_i(t)$  is a set of predicted GO terms for protein  $i$  using the threshold  $t$ , and  $T_i$  is a set of annotated GO terms for protein  $i$ .

We also use similar term-centric metrics to evaluate the model, as suggested by CAFA. However, instead of using receiver operating characteristic (ROC) curves, we choose to consider precision-recall (PR) curves and calculate the area under the curve. As pointed out by Davis and Goadrich [41], when dealing with highly imbalanced data sets, PR curves give a more informative picture of an algorithm's performance than ROC curves. Since this is a multi-label task, we adopt two averaged measures of area under the precision-recall curve (AUPR) for all terms:

$$pr_f(t) = \frac{\sum_i I(f \in P_i(t) \wedge f \in T_i)}{\sum_i I(f \in P_i(t))} \quad (11)$$

$$rc_f(t) = \frac{\sum_i I(f \in P_i(t) \wedge f \in T_i)}{\sum_i I(f \in T_i)} \quad (12)$$

$$\text{AUPR}_f = \sum_t (rc_f(t) - rc_f(t-1)) \cdot pr_f(t) \quad (13)$$

$$\text{M-AUPR} = \frac{1}{N_f} \cdot \sum_f \text{AUPR}_f \quad (14)$$

$$\text{m-AUPR} = \sum_t (rc(t) - rc(t-1)) \cdot pr(t) \quad (15)$$

Where  $pr_f$  and  $rc_f$  are precision and recall for a single GO term  $f$ ,  $\text{AUPR}_f$  is the area under the precision-recall curve (AUPR) for  $f$ , and  $N_f$  is the number of GO terms used for evaluation. The macro-averaged AUPR (M-AUPR) is defined as the unweighted mean of the AUPR for all labels, while the micro-averaged AUPR (m-AUPR) is calculated globally by considering each element of the label indicator matrix as a label.

### Comparison between different types of features

In this section, we compare the results of using different network information and node attributes. The purpose is to show which type of feature is most informative and how we can further improve the performance by combining multiple types of features. In Table 1, we show the results when using different network types and node attributes. For network types, we test among only using the PPI network, only using the SSN, and using the combined network. As for node attributes, we compare the performance among different attributes (sequence, protein domains, and subcellular location) and especially compare with ALL attributes (using all three kinds of attributes together). We do not consider sequence information as node attributes when the SSN is used as the network to train the model, because of the redundancy.

Looking at the performance of using different network types, the combined network is always better than the other two single network types across all three ontologies, which indicates that the PPI network and the SSN can provide complementary information for function prediction. Comparing the effect of the PPI network and the SSN, the PPI network shows better performance than the SSN, except on MF ontology. We can conclude that the PPI network provides more function-related information than the SSN, and that MF ontology has a great correlation with sequence information.

From the analysis of the effects of using different node attributes, we can conclude that the integration of all node attributes is better than using any single attribute across all three ontologies, no matter what kind of network type is used. We can see that using a sequence attribute can achieve reasonable performance for predicting functions in all three ontologies when combined with the PPI network, proving the importance of sequence information. It also shows that the protein domain attribute is the most informative feature for predicting MF functions, while the subcellular location attribute is important for CC function predictions. Overall, each type of node attribute contributes to the function prediction, and the combination of all features improves the performance and provides more robust predictions. Based on the results shown in Table 1, our final model considers both the SSN and PPI network as our network information. As for node attributes, we include subcellular location and protein domain features as node attributes in the SSN, and include all three types of features as node attributes in the PPI network.

### Comparison with BLAST

We first compare our method with a baseline model, BLAST, which is a widely used homology-based method for annotating protein functions [5]. We randomly split the data set into 80% for the training set and 20% for the test set. We compare our method with BLAST under two conditions: (1) the full data set, using all of the 80% training set; and (2) the partial data set, removing those training samples with more than 50% identity with one of the test samples (removing potential homologs). The training set is used for constructing the database. Then all GO term annotations of the best hit in the database are assigned to the query protein in the test set as the predicted functions. Since BLAST cannot assign probabilities for each prediction, we use Precision, Recall, and F1 score as the evaluation metrics instead of the M-AUPR, m-AUPR, and F-max we used in the previous section. For Graph2GO, we use the same training and test sets and use 0.5 as the threshold to assign predicted labels.

As shown in Table 2, in terms of all metrics, Graph2GO outperforms BLAST by a large margin across all three ontologies under both comparison conditions, especially for CC and BP, proving the superiority of our model. We can see that Graph2GO can achieve significantly higher Precision than BLAST under the full data set condition. BLAST results in many false positives and low Precision because of predicting functions simply based on amino acid sequence information but ignoring other informative features. Besides, it is shown that BLAST can obtain reasonable performance in MF ontology, which demonstrates that amino acid sequence information is highly related to the functions in MF ontology again. After removing highly similar sequences of the test set in the training set, the performances of both methods drop accordingly, while Graph2GO shows a smaller drop and thus outperforms BLAST even more under the partial data set condition. For BLAST, the performance drops from the full data set condition to the partial data set condition in terms of F1 score are 27%, 32%, and 37% for CC, MF, and BP, respectively. As for Graph2GO, the F1 score drops to 9%, 25%, and 17% for CC, MF, and BP, respectively, which are significantly lower scores than BLAST, exhibiting a more robust performance.

### Comparison with other network-based methods

To evaluate the performance of our method, we compare our method with two popular protein function prediction methods: Mashup and deepNF, which are also based on the concept of embedding. In their initial implementations, Mashup and deepNF use support vector machine (SVM) as the classification model. In order to make the performance more comparable, we also use DNN that is used by Graph2GO as their classification models. Each method is evaluated using 5-fold cross-validation, repeated 10 times. We first compare the performance using all GO terms, as shown in Fig. 3. We also group GO terms into three functional categories based on the sparsity levels for each ontology, each containing GO terms with 11-30, 31-100, and 101-300 proteins, and show the detailed comparisons in Supplementary Figures S2, S3, and S4.

For both CC and MF ontology, our model outperforms deepNF and Mashup in terms of m-AUPR and F-max, and achieves comparable results with deepNF and Mashup in terms of M-AUPR. Our model is comparable to Mashup and outperforms deepNF in BP ontology in terms of m-AUPR and F-max and is slightly worse than Mashup in M-AUPR. Looking at the detailed comparison in three sparsity levels shown in Supplementary Figure S2, our model can achieve the best performance consis-

**Table 1:** Performance comparison among using different network types (sequence similarity network, PPI network, and both) and node attributes (sequence, subcellular location, protein domains, and ALL)

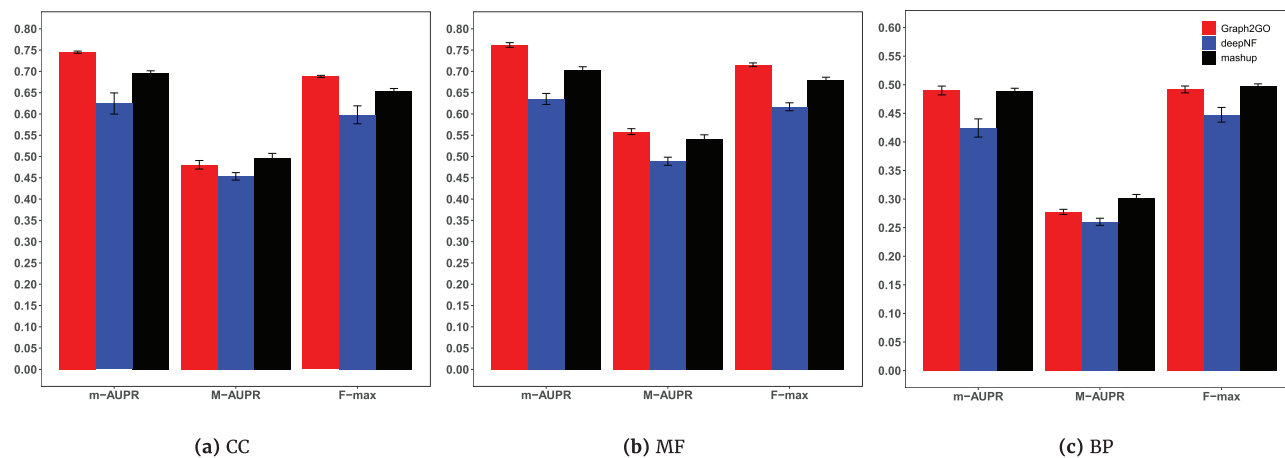
| Network  | Attribute | CC           |              |              | MF           |              |              | BP           |              |              |
|----------|-----------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
|          |           | M-AUPR       | m-AUPR       | F-max        | M-AUPR       | m-AUPR       | F-max        | M-AUPR       | m-AUPR       | F-max        |
| Sequence | location  | 0.325        | 0.686        | 0.639        | 0.411        | 0.650        | 0.624        | 0.152        | 0.336        | 0.388        |
|          | domain    | 0.217        | 0.594        | 0.568        | 0.383        | 0.669        | 0.639        | 0.144        | 0.357        | 0.384        |
|          | ALL       | 0.348        | 0.694        | 0.644        | 0.465        | 0.717        | 0.671        | 0.185        | 0.390        | 0.418        |
| PPI      | sequence  | 0.471        | 0.683        | 0.639        | 0.466        | 0.657        | 0.626        | 0.268        | 0.458        | 0.472        |
|          | location  | 0.448        | 0.704        | 0.652        | 0.398        | 0.596        | 0.567        | 0.233        | 0.435        | 0.450        |
|          | domain    | 0.443        | 0.681        | 0.634        | 0.476        | 0.678        | 0.635        | 0.258        | 0.464        | 0.470        |
| Combined | ALL       | 0.478        | 0.716        | 0.644        | 0.478        | 0.677        | 0.623        | 0.268        | 0.471        | 0.471        |
|          | location  | 0.465        | 0.744        | 0.686        | 0.499        | 0.715        | 0.672        | 0.239        | 0.448        | 0.463        |
|          | domain    | 0.426        | 0.695        | 0.643        | 0.520        | 0.751        | 0.698        | 0.253        | 0.465        | 0.473        |
|          | ALL       | <b>0.494</b> | <b>0.751</b> | <b>0.686</b> | <b>0.560</b> | <b>0.761</b> | <b>0.718</b> | <b>0.284</b> | <b>0.488</b> | <b>0.490</b> |

We use the M-AUPR, m-AUPR, and F-max as the evaluation metric. ALL means using the combination of all features as node attributes, and we do not include sequence as a node attribute when SSN is used to train the model.

**Table 2:** Performance comparison between BLAST and Graph2GO in terms of precision, recall, and F1 score under two conditions (full data set and partial data set after removing homologs)

| Method             | CC        |        |          | MF        |        |          | BP        |        |          |
|--------------------|-----------|--------|----------|-----------|--------|----------|-----------|--------|----------|
|                    | Precision | Recall | F1 Score | Precision | Recall | F1 Score | Precision | Recall | F1 Score |
| BLAST (full)       | 0.540     | 0.589  | 0.564    | 0.633     | 0.712  | 0.670    | 0.373     | 0.427  | 0.398    |
| BLAST (partial)    | 0.396     | 0.429  | 0.412    | 0.421     | 0.490  | 0.453    | 0.234     | 0.271  | 0.251    |
| Graph2GO (full)    | 0.766     | 0.624  | 0.688    | 0.816     | 0.639  | 0.717    | 0.686     | 0.354  | 0.467    |
| Graph2GO (partial) | 0.731     | 0.549  | 0.627    | 0.698     | 0.448  | 0.545    | 0.595     | 0.286  | 0.387    |

We use 0.5 as the threshold to assign predicted labels for Graph2GO. It should be noted that the F1 score here is different from the F-max used previously, since the threshold is prespecified for Graph2GO and there is no threshold for BLAST.

**Figure 3:** Performance comparisons with other network-based methods. Graph2GO is compared with Mashup and deepNF in terms of three metrics: micro-AUPR, macro-AUPR, and F-max. The figure shows the comparison results on (a) CC, (b) MF, and (c) BP ontology. Each method is evaluated using 5-fold cross-validation, repeated 10 times to calculate the confidence interval.

tently for MF ontology, across all sparsity levels. As we discussed above, the MF ontology has a great correlation with sequence information. Unlike Mashup and deepNF, which only consider PPI networks, Graph2GO explicitly includes sequence information in the model (SSN and sequence in the node attribute), which improves the performance of our model. As for CC ontology, Graph2GO can achieve the best performance for the most general categories (i.e., annotating between 101 and 300 proteins). For GO terms annotated between 11 and 100 proteins, our method is not as good as Mashup and deepNF. For BP ontology,

as the number of annotations per GO term increases, the performance of Graph2GO gradually improves and becomes comparable with other two methods.

It can be observed that Graph2GO works best for GO terms with enough annotated proteins, and is not as good as Mashup for very specific GO terms with few annotations for CC and BP ontology. In summary, our model can achieve state-of-the-art performance in both CC and MF ontology, especially as the number of annotated proteins increases, and can obtain comparable results in BP ontology. Compared with Mashup and deepNF, we

can conclude that both the inclusion of discriminating node attributes and the power of attribute propagation across two types of networks enable Graph2GO to obtain competing results.

### Performance on other species

In order to demonstrate the power and robustness of Graph2GO, we downloaded data from SwissProt and the STRING database for another 5 species (fruit fly, mouse, rat, *Saccharomyces cerevisiae*, and *Bacillus subtilis*), and tested the performance of Graph2GO on these species. For each species, we followed the same procedure as when we tested on a human data set, and the performance was evaluated in each species individually. Table 3 shows the results in terms of macro-AUPR, micro-AUPR, and F-max. We observe that Graph2GO can achieve consistently decent performance on all 5 species in terms of these evaluation metrics, which proves the robustness of our method. It should be noted that the performance on *Saccharomyces cerevisiae* and *Bacillus subtilis* is a lot better than on other species.

### Graph2GO web server

We also developed a web server based on Shiny app in R that supports protein function query on-the-fly: <https://integrativomics.shinyapps.io/graph2go/>. Currently, the web server supports 15,133 human proteins that are trained by our model. Network information (SSN and PPI network) and node attributes (subcellular location and protein domains) are displayed in our web server. The function prediction results are ranked by the probability score for each ontology separately, and users can specify the threshold to control the confidence of the results.

A promising application of our model is to utilize the latent representations to perform downstream analyses, since the generated representations already summarize various kinds of informative features. We support downstream clustering analysis for the query protein, where similarities are measured based on latent representations. Most similar proteins can be detected for the query protein to constitute a network module. A GO enrichment analysis can be performed on all proteins in the network module to analyze functions of the module. In the future, we are going to extend the server to include more proteins from other species.

## Discussion

### Feature transfer for solving annotation bias

A challenge in protein function prediction is the lack of annotated features for proteins, which are crucial for machine learning models. This is due to the bias in the experimental annotation of proteins and has impeded the development of biomedical research to some extent [42, 43]. In our experiment on the human data set, among 15,133 proteins, 5,716 proteins lack annotation of protein domains, while 2,364 lack annotation of subcellular location. Even with lots of missing values, when only using protein domains or subcellular location as node attributes, Graph2GO can still achieve reasonable results, as shown in Table 1. To demonstrate the power of Graph2GO to predict on these sparsely annotated proteins, we divide the test set into two groups: the sparsity group (without any subcellular location or protein domain annotation) and the non-sparsity group. We compare Graph2GO with a convolutional neural network (CNN) model. This CNN model uses the same node attributes as Graph2GO, but does not utilize networks to propagate pro-

tein attributes on these two groups separately. The detailed description of this CNN model is in the Supplementary Materials. The comparison results are shown in Supplementary Table S3. It is anticipated that the performance on the sparsity group should be worse than that of the non-sparsity group, due to the lack of features. We calculate the ratio between the performance of the sparsity and non-sparsity groups to measure how well Graph2GO and CNN handle the lack of features. The averaged ratios in terms of F-max for CNN are 74%, 44%, and 73% for CC, MF, and BP, respectively, while for Graph2GO, the averaged ratio are 101%, 84%, and 111%, respectively. We can see that when lacking features, the performance of CNN would drop significantly compared to the non-sparsity group. However, the performance of Graph2GO is much more stable when facing a lack of features. In some cases, the performance in the sparsity group is even better than the non-sparsity group. This is due to the feature transfer of Graph2GO, where node attributes are transferred between nodes in the graph to augment the information of each protein. Since proteins are connected in the interaction network, nodes without any attributes can still update their representations based on attributes of neighboring nodes, which solves the problem of missing values to some extent.

The encoding of original features into the model is of high importance to Graph2GO, not only for preserving information as much as possible, but also for feature transfer. We represent protein domains and subcellular location using the bag-of-words encoding method, which best preserves the original information and supports the addition of features that the feature transfer relies on. As for protein sequence, the CT encoding method loses sequential information and is not well suited for the addition operation. To solve this, we also utilize the similarity network to represent the sequence information, and we discover that the inclusion of both types of sequence representations is better than either of them alone, which might indicate that they can provide some complementary information.

### Extensibility and future work

Compared to other methods that also use multiple protein features [13, 14, 17], an advantage of our method is that it is convenient to incorporate other function-related information. Within the network architecture, all the features are regarded as node attributes and can be treated equally. In this paper, we found the integration of all these node features and the network information provides the best prediction performance. The model can be easily extended to take into account additional information, such as post-translational modifications, protein structure, or any other protein features. It's also worth incorporating more networks and finding a better way to integrate them. For example, we may use the seven channels in the STRING database separately instead of using the combined PPI network, as adopted by Mashup.

The architecture we proposed in this paper can not only be used to predict protein functions, but can also be applied to other tasks that involve biological networks, since the learned embeddings are informative and can be used as feature inputs for various downstream machine learning tasks. For example, one can predict the interactions between any two proteins, predict new therapeutic applications of an existing drug, or run a clustering algorithm to cluster genes into modules based on learned embeddings.

In principle, predicting GO terms is a hierarchical classification problem, since GO terms are organized in a hierarchy and are interacting with each other [44]. Currently, Graph2GO as-



**Table 3:** Performance on other 5 species in terms of macro-AUPR, micro-AUPR, and F-max metrics

| Species                         | CC     |        |       | MF     |        |       | BP     |        |       |
|---------------------------------|--------|--------|-------|--------|--------|-------|--------|--------|-------|
|                                 | M-AUPR | m-AUPR | F-max | M-AUPR | m-AUPR | F-max | M-AUPR | m-AUPR | F-max |
| Human                           | 0.494  | 0.751  | 0.686 | 0.560  | 0.761  | 0.718 | 0.284  | 0.488  | 0.490 |
| Fruit fly                       | 0.570  | 0.764  | 0.729 | 0.599  | 0.710  | 0.693 | 0.388  | 0.531  | 0.530 |
| Mouse                           | 0.480  | 0.743  | 0.681 | 0.583  | 0.763  | 0.710 | 0.262  | 0.482  | 0.483 |
| Rat                             | 0.461  | 0.701  | 0.652 | 0.577  | 0.743  | 0.699 | 0.267  | 0.429  | 0.448 |
| <i>Saccharomyces cerevisiae</i> | 0.677  | 0.848  | 0.781 | 0.597  | 0.740  | 0.689 | 0.481  | 0.647  | 0.620 |
| <i>Bacillus subtilis</i>        | 0.709  | 0.860  | 0.803 | 0.620  | 0.742  | 0.693 | 0.577  | 0.665  | 0.651 |

sumes the independence of all GO terms, which might cause inconsistencies between the prediction of leaf GO terms and the corresponding parent GO terms. In the future, we will consider adding some constraints on the output layer of our classification model to force the prediction to be consistent between leaf nodes and parent nodes. We will also try to incorporate the GO hierarchy into the model as 1 of the network structures to inform the training process.

## Conclusion

In this work, we present Graph2GO, a graph-based deep learning model that can make use of heterogeneous information in a unified way to predict protein functions. Our network representation learning method can take into account both network structure and protein attributes, to make predictions by organizing proteins into an attributed network architecture. Graph2GO achieves state-of-the-art performance on the benchmark data set and can easily be adapted to solve other tasks involving biological networks, such as link prediction, node classification, and sub-network discovery.

## Supplementary Data

There is an additional Supplementary Materials file containing additional information. It includes the following figures and tables:

**Supplementary Figure 1.** Cross-validation results of choosing different number of layers for the GCN model. The performances on all three ontologies are best when using two layers of graph convolutional operators. As the number of layers becomes larger the performance gets worse, because applying a graph convolutional operator repeatedly may mix node features from different clusters and make them indistinguishable.

**Supplementary Figure 2.** Performance comparison with other state-of-the-art methods in CC ontology for three different sparsity levels. Graph2GO is compared with Mashup and deepNF in terms of three metrics: micro-AUPR, macro-AUPR, and F-max. There, three figures show the results of 3 sparsity levels: [11-30], [31-100], and [101-300]. Each method is evaluated using 5-fold cross validation, repeated 10 times to calculate the confidence interval.

**Supplementary Figure 3.** Performance comparison with other state-of-the-art methods in MF ontology for three different sparsity levels. Graph2GO is compared with Mashup and deepNF in terms of three metrics: micro-AUPR, macro-AUPR, and F-max. There, three figures show the results of three sparsity levels: [11-30], [31-100], and [101-300]. Each method is evaluated using 5-fold cross validation, repeated 10 times to calculate the confidence interval.

**Supplementary Figure 4.** Performance comparison with other state-of-the-art methods in BP ontology for three different sparsity levels. Graph2GO is compared with Mashup and deepNF in terms of three metrics: micro-AUPR, macro-AUPR, and F-max. There, three figures show the results of three sparsity levels: [11-30], [31-100], and [101-300]. Each method is evaluated using 5-fold cross validation, repeated 10 times to calculate the confidence interval.

**Supplementary Table 1.** Classification of amino acids according to their dipoles and volumes of the side chains based on the CT method.

**Supplementary Table 2.** Performance between the model where we train an independent VGAE for each network and combine their embeddings (referred to as “individual”) and where we first combine the networks and train one VGAE to obtain the overall embedding (referred to as “combined”) in terms of M-AUPR, m-AUPR, and F-max. The experiment procedure is the same as other experiments performed in the “Results” section and is described there.

**Supplementary Table 3.** Performance of Graph2GO and CNN on sparsity group and non-sparsity group of the test set in terms of F1 score. Here, “sp” means the sparsity group, while “non” means the non-sparsity group.

## Availability of source code and requirements

- Project name: Graph2GO
- Project home page: <https://github.com/yanzhanglab/Graph2GO>
- Operating system(s): Platform independent
- Programming language: Python, R
- Other requirements: not applicable
- License: MIT
- RRID: SCR\_018726
- biotoolsID: biotools:graph2go

## Availability of supporting data and materials

All data are publicly available in our Github project home page. An archival copy of the code and supporting data are available via the GigaScience database, GigaDB [45].

## Abbreviations

AUPR: area under the precision-recall curve; BLAST: Basic Local Alignment and Search Tool; BP: biological process; CAFA: Critical Assessment of protein Function Annotation algorithms; CC: cellular component; CNN: convolutional neural network; CT: conjoint triad; DNN: deep neural network; GCN: graph convolutional network; GO: gene ontology; GPU: graphics processing

unit; MF: molecular function; M-AUPR: macro-averaged area under the precision-recall curve; m-AUPR: micro-averaged area under the precision-recall curve; PPI: protein-protein interaction; ROC: receiver operating characteristic; SSN: sequence similarity network; SVM: support vector machine; VAE: variational auto-encoder; VGAE: variational graph auto-encoder.

## Competing Interests

Y.G. serves as a consultant at Eli Lilly and Company and a consultant at Merck Group.

## Funding

This work was partially supported by The Ohio State University Startup Funds to Y.Z.

## Authors' contributions

Y.Z. and Y.G. conceived the project and supervised the study. K.F. performed the analysis and developed the tools. K.F. and Y.Z. drafted the manuscript. All authors read and approved the final manuscript.

## Acknowledgements

We acknowledge the use of the compute resources of the Ohio Supercomputer Center.

## REFERENCES

1. Radivojac P, Clark WT, Oron TR, et al. A large-scale evaluation of computational protein function prediction. *Nat Methods* 2013;**10**(3):221–7.
2. Loewenstein Y, Raimondo D, Redfern OC, et al. Protein function annotation by homology-based inference. *Genome Biol* 2009;**10**(2):207.
3. Piovesan D, Luigi Martelli P, Fariselli P, et al. BAR-PLUS: the Bologna Annotation Resource Plus for functional and structural annotation of protein sequences. *Nucleic Acids Res* 2011;**39**(Suppl 2):W197–202.
4. Chitale M, Hawkins T, Park C, et al. ESG: extended similarity group method for automated protein function prediction. *Bioinformatics* 2009;**25**(14):1739–45.
5. Jones CE, Baumann U, Brown AL. Automated methods of predicting the function of biological sequences using GO and BLAST. *BMC Bioinformatics* 2005;**6**(1):272.
6. Sharan R, Ulitsky I, Shamir R. Network-based prediction of protein function. *Mol Syst Biol* 2007;**3**(1):88.
7. Chua HN, Sung WK, Wong L. Using indirect protein interactions for the prediction of gene ontology functions. *BMC Bioinformatics* 2007;**8**(Suppl 4):S8.
8. Forslund K, Sonnhammer EL. Predicting protein function from domain content. *Bioinformatics* 2008;**24**(15):1681–7.
9. Rentzsch R, Orengo CA. Protein function prediction using domain families. *BMC Bioinformatics* 2013;**14**(Suppl 3):S5.
10. Lee D, Redfern O, Orengo C. Predicting protein function from sequence and structure. *Nat Rev Mol Cell Biol* 2007;**8**(12):995–1005.
11. Jensen LJ, Gupta R, Blom N, et al. Prediction of human protein function from post-translational modifications and localization features. *J Mol Biol* 2002;**319**(5):1257–65.
12. Verspoor KM. Roles for text mining in protein function prediction. *Methods Mol Biol* 2014;**1159**:95–108.
13. Piovesan D, Giollo M, Leonardi E, et al. INGA: protein function prediction combining interaction networks, domain assignments and sequence similarity. *Nucleic Acids Res* 2015;**43**(W1):W134–40.
14. Zhang C, Freddolino PL, Zhang Y. COFACTOR: improved protein function prediction by combining structure, sequence and protein–protein interaction information. *Nucleic Acids Res* 2017;**45**(W1):W291–9.
15. Boutet E, Lieberherr D, Tognolli M, et al. UniProtKB/Swiss-Prot, the manually annotated section of the UniProt KnowledgeBase: how to use the entry view. *Methods Mol Biol* 2016;**1374**:23–54.
16. Szklarczyk D, Morris JH, Cook H, et al. The STRING database in 2017: quality-controlled protein–protein association networks, made broadly accessible. *Nucleic Acids Res* 2017;**45**(D1):D362–8.
17. Kulmanov M, Khan MA, Hoehndorf R. DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics* 2018;**34**(4):660–8.
18. Kearnes S, McCloskey K, Berndl M, et al. Molecular graph convolutions: moving beyond fingerprints. *J Comput Aided Mol Des* 2016;**30**(8):595–608.
19. Backstrom L, Leskovec J. Supervised random walks: predicting and recommending links in social networks, United States Conference: WSDM'11: Fourth ACM International Conference on Web Search and Data Mining Hong Kong China February, 2011. p. 635–44, Association for Computing Machinery: New York, NY.
20. Duvenaud DK, Maclaurin D, Iparraguirre J, et al. Convolutional networks on graphs for learning molecular fingerprints. *Adv Neural Inf Process Syst* 28 (NIPS 2015):2224–32. <https://papers.nips.cc/paper/5954-convolutional-networks-on-graphs-for-learning-molecular-fingerprints>.
21. Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs. *Adv Neural Inf Process Syst* 30 (NIPS 2017):1024–34. <https://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs>.
22. Cho H, Berger B, Peng J. Compact integration of multi-network topology for functional analysis of genes. *Cell Syst* 2016;**3**(6):540–8.
23. Gligorijević V, Barot M, Bonneau R. deepNF: deep network fusion for protein function prediction. *Bioinformatics* 2018;**34**(22):3873–81.
24. Bhagat S, Cormode G, Muthukrishnan S. Node classification in social networks. In: *Social network data analytics*. 2011;115–48, Springer.
25. Vishwanathan SVN, Schraudolph NN, Kondor R, et al. Graph kernels. *J Mach Learn Res* 2010;**11**(Apr):1201–42.
26. Hamilton WL, Ying R, Leskovec J. Representation learning on graphs: methods and applications, *IEEE Data Engineering Bulletin*. 2017;**40** (3):52–74.
27. Ashburner M, Ball CA, Blake JA, et al. Gene ontology: tool for the unification of biology. *Nat Genet* 2000;**25**(1):25–9.
28. Altschul SF, Madden TL, Schäffer AA, et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* 1997;**25**(17):3389–402.
29. Shen J, Zhang J, Luo X, et al. Predicting protein–protein interactions based only on sequences information. *Proc Natl Acad Sci USA* 2007;**104**(11):4337–41.

30. Muppirlala UK, Honavar VG, Dobbs D. Predicting RNA-protein interactions using only sequence information. *BMC Bioinformatics* 2011;12(1):489.
31. You ZH, Lei YK, Zhu L, et al. Prediction of protein-protein interactions from amino acid sequences with ensemble extreme learning machines and principal component analysis. *BMC Bioinformatics* 2013;14 (Suppl 8):S10.
32. Wang H, Hu X. Accurate prediction of nuclear receptors with conjoint triad feature. *BMC Bioinformatics* 2015;16(1):402.
33. Wang J, Zhang L, Jia L, et al. Protein-protein interactions prediction using a novel local conjoint triad descriptor of amino acid sequences. *Int J Mol Sci* 2017;18(11):2373.
34. Kipf TN, Welling M. Variational graph auto-encoders, 2016. Bayesian Deep Learning Workshop (NIPS 2016), arXiv preprint ([arXiv:161107308](https://arxiv.org/abs/161107308)).
35. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks, 2016. arXiv preprint ([arXiv:160902907](https://arxiv.org/abs/160902907)). ICLR 2017.
36. Kingma DP, Welling M. Auto-encoding variational Bayes, 2013. arXiv preprint ([arXiv:13126114](https://arxiv.org/abs/13126114)).
37. Li Q, Han Z, Wu XM. Deeper insights into graph convolutional networks for semi-supervised learning. In: Programs and Abstracts of the Thirty-Second AAAI Conference on Artificial Intelligence; 2018. p. 3538–45, AAAI, New Orleans, Louisiana, USA.
38. LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature* 2015;521(7553):436–44.
39. Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks, Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 13–15 May, 2010. vol. 9. p. 249–56, Chia Laguna Resort, Sardinia, Italy.
40. Kingma DP, Ba J. Adam: a method for stochastic optimization, Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. arXiv preprint ([arXiv:14126980](https://arxiv.org/abs/1412.6980)).
41. Davis J, Goadrich M. The relationship between precision-recall and ROC curves, ICML '06: Proceedings of the 23rd international conference on Machine learning June, 2006, p. 233–40, Association for Computing Machinery: Pittsburgh, Pennsylvania.
42. Haynes WA, Tomczak A, Khatri P. Gene annotation bias impedes biomedical research. *Sci Rep* 2018;8(1):1362.
43. Schnoes AM, Ream DC, Thorman AW, et al. Biases in the experimental annotations of protein function and their effect on our understanding of protein function space. *PLOS Comput Biol* 2013;9(5):e1003063.
44. Kahanda I, Funk CS, Ullah F, et al. A close look at protein function prediction evaluation protocols. *GigaScience* 2015;14(4):41.
45. Fan K, Guan Y, Zhang Y. Supporting data for “Graph2GO: a multi-modal attributed network embedding method for inferring protein functions”, 2020. <http://dx.doi.org/10.5524/100761>.