Supplementary information for "Modeling multi-scale data via a network of networks"

Shawn Gu[1], Meng Jiang[1], Pietro Hiram Guzzi[2], and Tijana Milenković[1, *]

[1]Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA
[2]Department of Surgical and Medical Sciences, University Magna Graecia of Catanzaro, Catanzaro, IT
[*]To whom correspondence should be addressed (email: tmilenko@nd.edu).

S1 Materials and methods

S1.1 Data

S1.1.1 Our synthetic NoN generator

Let M be the set of random graph models we consider for generating the synthetic NoN. For random graph model $m \in M$, let $m(x, y)$ be a random graph of type m with x nodes and y edges. Let $M^{(2)} = M \times M$ be the set of all possible combinations of the elements of M with themselves. Let $|V^{(2)}|$ be the target number of nodes at level 2, $|E^{(2)}|$ be the target number of edges at level 2, $|V^{(1)}|$ be the target number of nodes for each network at level 1, and $|E^{(1)}|$ be the target number of edges for each network at level 1; these parameters allow us to generate synthetic NoNs that approximate the size of real-world NoNs. Note that in our synthetic NoN generation, we fix the size of the level 1 networks to eliminate any effect of level 1 network size; however, our model can easily generate level 1 networks of varying size.

For each $(m_1, m_2) \in M^{(2)}$, we generate k isolated NoN regions where, for each region, the level 2 network is of type m_2 and every level 1 network is of type m_1. This results in $k|M^{(2)}|$ total isolated NoN regions. After combining all of them, the resulting NoN should have $|V^{(2)}|$ nodes and $|E^{(2)}|$ edges. As such, for each $(m_1, m_2) \in M^{(2)}$, we generate k isolated NoN regions

$$\begin{aligned} \{G^{(2)}_{(m_1,m_2)} &= m_2(\lfloor \frac{|V^{(2)}|}{k|M^{(2)}|} \rfloor, \lfloor \frac{|E^{(2)}|}{k|M^{(2)}|} \rfloor) \text{ and} \\ G^{(1)}_{(m_1,m_2)_i} &= m_1(|V^{(1)}|, |E^{(1)}|) \text{ for } i \in \{1, ..., \lfloor \frac{|V^{(2)}|}{k|M^{(2)}|} \rfloor\}\}. \end{aligned} \tag{1}$$

Because real-world systems are likely to have many groups of nodes, we set $k = 5$ for our synthetic NoNs, corresponding to five instances of each of the four random graph model combinations. Then, we connect these isolated NoN regions by randomly removing edges within level 2 node groups and randomly adding the same number of edges across level 2 nodes groups (*across-edge* amount). Specifically, we repeat the following process $a\% \times |E^{(2)}|$ times: (i) randomly select a level 2 node group, (ii) randomly select an edge in that node group, (iii) delete that edge, (iv) randomly select two level 2 nodes from different node groups, and (v) add an edge between the selected nodes. If the resulting NoN is still disconnected, we redo the process with a different random seed. While we could impose a condition to guarantee connectedness, doing so would bias the generation. If a connected NoN can not be found after 10 tries, we just continue with the last one. We start with $a = 5$ to retain most of the level 2 node groups' original GEO- and SF-like network topologies, and we vary a to be 25, 50, 75, and 95 to test the effect of breaking the network topologies down. This also means that at $a = 5$ there is significant clustering (each level 2 node group consists of densely interconnected nodes), while at $a = 95$ there is very little clustering.

We also introduce random rewiring to test each method's robustness to data noise (*rewire-noise* amount). Specifically, for r% rewire-noise, for each level 1 network, we randomly delete $\frac{r}{100} \times |E^{(1)}|$ edges and randomly add the same number back. For the level 2 network, for each node group, we randomly delete r% of $\lfloor \frac{|E^{(2)}|}{5|M^{(2)}|} \rfloor$ edges and randomly add the same number back. We vary r to be 0 (no noise), 10, 25, 50, 75, 100 (completely random).

S1.1.2 PIN-PSN NoN

We construct a biological NoN using the human PIN and the proteins' associated PSNs. We obtain human PPI data from BioGrid [9] version 4.1.190. We keep only physical interactions, remove selfloops and multi-edges, and take the largest connected component. This results in a final size of 18,708 nodes and 434,527 edges.

We map proteins in our PIN to their corresponding PDB IDs as follows. Considering the proteins' BioGrid IDs, we use UniProt's [2] mapping service (version 2020_06) to obtain BioGrid-to-UniProt mappings. Any mappings that are not reviewed (i.e., not Swiss-Prot) are discarded. Next, we remove any mapped data when more than one BioGrid ID is mapped to a UniProt ID and vice versa, leaving only one-to-one mappings between BioGrid IDs and UniProt IDs. Then, we repeat the process starting with the proteins' official symbol IDs. As such, for each protein, we have two UniProt IDs: one originating from its BioGrid ID and the other from its official symbol ID. To remove any ambiguity moving forward, we only keep proteins whose two UniProt IDs are equal. In total, we have 16,079 such UniProt IDs.

Given these UniProt IDs, we again use UniProt's mapping service, but this time to map UniProt IDs to PDB IDs. Then, we remove any PDB ID whose PDB structure has a resolution greater than or equal to 3.0Å, as PDB considers these to be "low resolution" [8]. Next, to obtain a one-to-one mapping between Uniprot IDs and PDB IDs, we form one set out of every protein sequence associated with the UniProt IDs and another set out of every protein sequence associated with every PDB chain (each PDB ID can have multiple corresponding chains). We perform all-vs-all protein sequence comparison using BLASTP [1] between these two sets and take only reciprocal best hits as our final one-to-one UniProt-to-PDB mappings. After this step, we have 4,776 PDB chains.

Regarding GO term labels, we only consider those GO terms with 20 or more positive instances to ensure there is enough data to perform classification on.

S1.2 Existing approaches for label prediction

Recall that we consider graph theoretic approaches based on graphlets and graph learning approaches, namely, SIGN and DiffPool.

Graphlets are small subgraphs (a path, triangle, square, etc.) that can be considered the building blocks of networks, and they can be used to extract features of both nodes and networks. For each node in a general network, for each automorphism orbit (intuitively, node symmetry group) in a graphlet, one can count the number of times the node is a part of a given graphlet orbit. These counts are summarized into the node's feature, also called its *graphlet degree vector* (GDV); when considering up to 4-node graphlets, GDVs will have length 15. Then, to extract features of the entire network, GDVs of all nodes can be collected into the network's *GDV matrix* (GDVM) feature. One drawback of the GDVM feature is that its dimensions depend on the number of nodes in the network – if performing graph classification of different sized networks using GDVM features, issues can arise. Thus, we also consider a transformation of the GDVM, the graphlet correlation matrix (GCM) [10], which always has the same dimensions regardless of network size.

Given these definitions of graphlet features for nodes in a general network or for the entire general network itself, we now explain which features we use for nodes in a level 2 network and which features we use for level 1 networks. For the former, we extract each level 2 node's GDV (L2 GDV). For the latter, we extract each level 1 network's GDVM and GCM (L1 GDVM and L1 GCM). We use L1 GDVM when analyzing synthetic NoNs since we found that it outperformed L1 GCM. For the biological NoN, L1 GCM is the only viable feature since level 1 networks (PSNs) have different numbers of nodes (amino acids).

Then, to obtain NoN graphlet features, we concatenate level 2 nodes' L2 GDVs with their networks' L1 GDVMs or L1 GCMs. This results in five graphlet-based features: those for level 1 networks (L1 GDVM and L1 GCM) that are used for graph label prediction, those for nodes in a level 2 network (L2 GDV) that

are used for node label prediction, and those for the entire NoN (L1 GDVM + L2 GDV and L1 GCM + L2 GDV) that are used for entity label prediction. In order to perform classification, for each graphlet-based feature, we train a logistic regression classifier (Supplementary Section S1.4). So for example, when we say L2 GDV, we mean the L2 GDV feature under logistic regression.

SIGN consists of two parts. First, it extracts different types of adjacency matrices from a network. SIGN specifically considers the traditional adjacency matrix, the Personalized PageRank-based adjacency matrix [6], the triangle-induced adjacency matrix [7], and their powers (see Supplementary Section S1.4 for which powers are used); these matrices are concatenated row-wise. Second, they are given as input into a neural network classifier. Mathematically, SIGN overall is equivalent to an ensemble of multiple one-layer-deep (i.e., shallow) GCN classifiers, which is why it is considered a graph learning approach.

DiffPool aims to perform graph classification. However, unlike graphlet-based approaches and SIGN, which extract "general purpose" features of nodes/networks that can be used in any downstream machine learning task (label prediction in our study), DiffPool does not extract general purpose features. Instead, for each input network, given initial features for each node, DiffPool uses a GNN to aggregate the nodes' initial features into a summary hidden feature for the entire network. Then, given hidden features corresponding to the input networks, the GNN is trained to perform graph classification. Since the GNN is trained over many iterations, the hidden feature is dependent on the training data and can only be used as a part of DiffPool's GNN. When we say L1 DiffPool, we mean its GNN with the initial features chosen (Supplementary Section S1.4), for graph classification using only level 1 networks.

As SIGN and DiffPool are single-level graph learning approaches, we also combine them into an NoN graph learning approach. Given each level 2 node's feature extracted by SIGN, we concatenate it with the level 2 node's corresponding level 1 network's hidden feature computed by DiffPool's GNN. The GNN is then trained on these concatenated features to perform classification (note that any general purpose feature can be incorporated into DiffPool like this). When we say L1 DiffPool + L2 SIGN, we mean entity label prediction using the process described above, incorporating SIGN's extracted feature into DiffPool's GNN. So, we use three graph learning-based approaches: L1 DiffPool, L2 SIGN, and L1 DiffPool + L2 SIGN.

We also combine L1 GDVM + L2 GDV or L1 GCM + L2 GDV with L1 DiffPool + L2 SIGN to test whether integrating information across the graph theoretic and graph learning domains improves upon either alone. Graphlet-based features can be incorporated into DiffPool using the process described previously.

In total, we have five single-level approaches: L1 GDVM, L1 GCM, L2 GDV, L1 DiffPool, and L2 SIGN; and five NoN approaches: L1 GDVM + L2 GDV, L1 GCM + L2 GDV, L1 DiffPool + L2 SIGN, L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN, and L1 GCM + L2 GDV + L1 DiffPool + L2 SIGN.

Finally, note that we did test node2vec [4], a prominent random walk-based embedding method, as a graph learning approach. node2vec extracts general purpose features like graphlets and SIGN, so we used it with logistic regression. However, DiffPool outperformed node2vec in level 1 graph classification, SIGN outperformed node2vec in level 2 node classification, and L1 DiffPool + L2 SIGN outperformed any combination involving node2vec in level 2 node classification for the entire NoN.

S1.3 Our integrative GCN approach

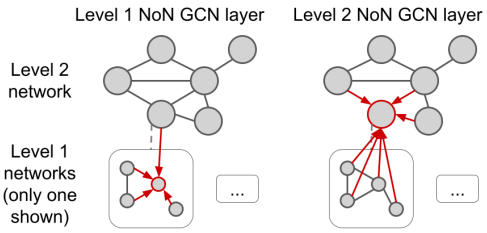
Here, we describe how we generalize GCNs to apply to NoNs. First, we summarize basic GCNs. Second, we discuss our extensions.

The important unit of a GCN is the graph convolutional layer, which works as follows. For each node in some network $G = (V, E)$, the node's features are aggregated with its neighbors' features and then these aggregated features are propagated to the next layer of the neural network. More formally, summarized from [5], let A be the adjacency matrix of G and H be a $|V| \times d$ matrix of G's nodes' features at the current layer (the i^{th} row corresponds to the feature of the i^{th} node). Then, forward propagation is carried out through

$$f(H, A) = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}HW), \tag{2}$$

where W is the trainable weight matrix for the current layer, σ is an activation function, $\tilde{A} = A + I$ is the adjacency matrix with self-loops added (so that mathematically, the aggregation actually includes each node's features along with its neighbors features) and $\tilde{D}$ is the diagonal node degree matrix used for normalizing the adjacency matrix.

Essentially, graph convolutions allow each node to see information about its neighbors. So, what if we generalized graph convolutions to NoNs so that each node sees information not only about its neighbors (in the same level), but also about its corresponding network at a lower level or about the network it is a part of at a higher level? This would be in line with our intuition that the feature of a protein should contain information about how it interacts with other proteins (i.e., its topology in the level 2 network) and properties of the protein itself that allow for such interactions (topology of level 1 nodes in its level 1 network). So, we define a two part NoN-GCN layer consisting of one part that propagates level 2 nodes and another part that propagates level 1 nodes that attempts to do this (below and Fig. S1).



Supplementary Figure S1: Illustration of the two part NoN-GCN layer. In the level 1 NoN-GCN layer, the level 1 node circled in red receives features from its neighbors in its level 1 network as well as features of the level 2 node its level 1 network corresponds to. This is done for every level 1 node in every level 1 network. In the level 2 NoN-GCN layer, the level 2 node circled in red received features from its neighbors in the level 2 network as well as features of each of the level 1 nodes in its level 1 network. This is done for every level 2 node in the level 2 network.

Let $\{G^{(2)} = (V^{(2)}, E^{(2)})$ and $\{G_1^{(1)}, ..., G_{|V^{(2)}|}^{(1)}\}\}$ be an NoN. Let $A^{(2)}$ be the adjacency matrix of $G^{(2)}$. Let ${}^kH^{(2)}$ be the $|V^{(2)}| \times d$ matrix of features for $G^{(2)}$ after the k^{th} neural network layer; for $k = 0$, this would correspond to the input feature matrix (for example, $G^{(2)}$'s GDVM or GCM). Let ${}^kh_i^{(2)}$ be the feature vector of the i^{th} node $v_i^{(2)} \in V^{(2)}$ (for example, $v_i^{(2)}$'s GDV). Let $A_i^{(1)}$ be the adjacency matrix of $v_i^{(2)}$'s level 1 network $G_i^{(1)}$. Let ${}^kH_i^{(1)}$ be the $|V_i^{(1)}| \times d$ matrix of features for $G_i^{(1)}$ after the k^{th} neural network layer. Let ${}^kh_{i_j}^{(1)}$ be the feature vector of the j^{th} node $v_{i_j}^{(1)}$ of $G_i^{(1)}$.

Propagation at level 2 works as follows. For each node $v_i^{(2)}$, for the feature matrix ${}^kH_i^{(1)}$ of its corresponding level 1 network $G_i^{(1)}$, we take the average over all of ${}^kH_i^{(1)}$'s rows to obtain a $1 \times d$ vector as a "summary" feature vector of $G_i^{(1)}$. Then, we combine these resulting vectors over all level 1 networks into a $|V^{(2)}| \times d$ matrix ${}^k\bar{H}^{(2)}$, where each row corresponds to a level 1 network. In other words, ${}^k\bar{H}^{(2)}$ can be thought of as the feature matrix of the level 2 network based on each node's level 1 network (whereas ${}^kH^{(2)}$ is the feature matrix of the level 2 network based on each level 2 node). Then, our level 2 NoN-GCN layer forward propagation is carried out through

$$\begin{aligned} {}^{k+1}H^{(2)} = {}^{k+1}f_{l2}({}^kH^{(2)}, {}^k\bar{H}^{(2)}, A^{(2)}) = \\ \sigma(\tilde{D^{(2)}}^{-\frac{1}{2}}\tilde{A^{(2)}}\tilde{D^{(2)}}^{-\frac{1}{2}} \\ ({}^kH^{(2)} + {}^k\bar{H}^{(2)})^{k+1}W^{(2)}), \end{aligned} \tag{3}$$

where ${}^{k+1}W^{(2)}$ is the trainable weight matrix for the level 2 NoN-GCN layer, σ is an activation function, $\tilde{A} = A + I$ is the adjacency matrix with self-loops added and $\tilde{D}$ is the diagonal node degree matrix used for normalizing the adjacency matrix.

Propagation at level 1 works as follows. For each node $v_{i_j}^{(1)}$ in each level 1 network $G_i^{(1)} = (V_i^{(1)}, E_i^{(1)})$, we sum its feature with all of its neighbors' features as well as the feature of $G_i^{(1)}$'s corresponding level 2 node. Mathematically, this corresponds to the following for each level 1 network. Let ${}^k\bar{H}_i^{(1)}$ be a $|V_i^{(1)}| \times d$ matrix consisting of ${}^kh_i^{(2)}$ repeated $|V_i^{(1)}|$ times. This can be thought of as the (naive) feature matrix of

the level 1 network based on its corresponding level 2 node. Importantly ${}^{k}\bar{H}_i^{(1)}$ has the same dimensions as ${}^{k}H_i^{(1)}$. Then, level 1 NoN-GCN layer forward propagation is carried out for one level 1 network through

$$\begin{aligned} {}^{k+1}H_i^{(1)} &= {}^{k+1}f_{l1_i}({}^{k}\bar{H}_i^{(1)}, {}^{k}H_i^{(1)}, A_i^{(1)}) \\ &= \sigma(\tilde{D^{(1)}}_i^{-\frac{1}{2}} \tilde{A^{(1)}}_i \tilde{D^{(1)}}_i^{-\frac{1}{2}} \\ &\quad ({}^{k}H_i^{(1)} + {}^{k}\bar{H}_i^{(1)})^{k+1}W_i^{(1)}), \end{aligned} \tag{4}$$

where ${}^{k+1}W_i^{(1)}$ is the trainable weight matrix for the i^{th} level 1 network for the current level 1 NoN-GCN layer, σ is an activation function, $\tilde{A} = A + I$ is the adjacency matrix with self-loops added and $\tilde{D}$ is the diagonal node degree matrix used for normalizing the adjacency matrix.

So, one full NoN-GCN layer takes as input

$$\begin{aligned} &{}^{k}H^{(2)}, {}^{k}\bar{H}^{(2)}, A^{(2)}, \\ &\{{}^{k}\bar{H}_1^{(1)}, ..., {}^{k}\bar{H}_{|V^{(2)}|}^{(1)}\}, \\ &\{{}^{k}H_1^{(1)}, ..., {}^{k}H_{|V^{(2)}|}^{(1)}\}, \\ &\text{and } \{A_1^{(1)}, ..., A_{|V^{(2)}|}^{(1)}\}, \end{aligned} \tag{5}$$

and returns ${}^{k+1}H^{(2)}$ and $\{{}^{k+1}H_1^{(1)}, ..., {}^{k+1}H_{|V^{(2)}|}^{(1)}\}$. These outputs can then be fed as inputs (along with ${}^{k+1}\bar{H}^{(2)}$, which can be constructed from $\{{}^{k+1}H_1^{(1)}, ..., {}^{k+1}H_{|V^{(2)}|}^{(1)}\}$, and each ${}^{k+1}\bar{H}_i^{(1)}$, which can be constructed from its corresponding ${}^{k+1}h_i^{(2)}$) into another NoN-GCN layer, thus allowing these layers to be chained.

We refer to a GCN approach using λ layers as "GCN-λ".

Note that our implementation of the above is based on the `spektral` GNN library [3].

S1.4 Evaluation

For a given NoN $\{G^{(2)} = (V^{(2)}, E^{(2)})$ and $\{G_1^{(1)}, ..., G_{|V^{(2)}|}^{(1)}\}\}$, its label set $Y = y_1, ..., y_c$, and a function that maps level 2 nodes to their true labels $f_{true} : V^{(2)} \rightarrow Y$, the goal is to learn a predictive function $f_{pred} : V^{(2)} \rightarrow Y$. We do this by first splitting the data into three disjoint sets: training ($V_{tr}^{(2)}$), validation ($V_{val}^{(2)}$), and testing ($V_{te}^{(2)}$). Then, we train a classifier on the training set that aims to minimize the cross-entropy loss between $f_{true}(V_{tr}^{(2)})$ and $f_{pred}(V_{tr}^{(2)})$. We using $V_{val}^{(2)}$ to optimize hyperparameters and finally report the classifier's performance on $V_{te}^{(2)}$. Details are as follows.

Denote $Y = y_1, ..., y_c$ to be the set of possible level 2 node labels (recall for synthetic NoNs, given m random graph models, multiclass classification is done on $m \times m$ labels; for the real-world NoN, for each of the 131 ground truth datasets, binary classification is done on whether proteins have the corresponding label or not) and $f_{true} : V^{(2)} \rightarrow Y$ to be a function that maps level 2 nodes to their true labels. We split the set of level 2 nodes $V^{(2)}$ into three disjoint subsets as follows. p% of the data is randomly removed from $V^{(2)}$ and put into the training set $V_{tr}^{(2)}$. Half of the data remaining from $V^{(2)}$ is randomly removed and put into the validation set $V_{val}^{(2)}$. The remaining data is put into the testing set $V_{te}^{(2)}$. This results in three sets with size ratio $p{:}\frac{1-p}{2}{:}\frac{1-p}{2}$. Importantly, this splitting is done with the constraint that the distribution of node labels in each of the three sets matches the original label distribution of $V^{(2)}$ as closely as possible (i.e., stratified sampling). We train the classifier on $V_{tr}^{(2)}$, optimize hyperparameters using $V_{val}^{(2)}$, and report results on $V_{te}^{(2)}$. We repeat the random data splitting 3 different times and perform classification for each, reporting the average results over them. We do this 3 times so that 1) the effect of randomness from sampling reduced and 2) running the the approaches is still computationally feasible. For synthetic NoNs, we choose $p = 0.8$ (corresponding to a 8:1:1 data ratio), as this is a common split amount when data is not scarce. For real-world NoNs, we choose $p = 1/3$ (corresponding to a 1:1:1 data ratio). Because some of the ground truth

sets have as few as 20 positive instances, larger values of p would result in the validation/testing sets having too few of them.

Below, we describe classifier details. For graphlet-based approaches, we use each of L1 GDVM, L1 GCM, L2 GDV, L1 GDVM + L2 GDV, and L1 GCM + L2 GDV in logistic regression. For L2 SIGN, we use its features in own classifier. We refer to these as "regular classification". For approaches involving DiffPool, we run them as described in Supplementary Section S1.2. We refer to these as "DiffPool-based classification". Finally, we refer to classification using NoN-GCNs as "NoN-GCN-based" classification.

For a given data split, for each feature we consider in regular classification, we train the corresponding classifier using the ADAM optimizer on $V_{tr}^{(2)}$. We test the following learning rates $\{0.1, 0.01, 0.001\}$ and choose the best one with respect to performance when predicting on $V_{val}^{(2)}$. Then, we use this best classifier to predict on $V_{te}^{(2)}$.

For a given data split, for DiffPool-based classification, we perform a grid search over the following hyperparameters: `hidden dimension:` $\{32, 64, 128\}$ and `output dimension:` $\{32, 64, 128\}$. We choose the best combination with respect to performance when prediction on $V_{val}^{(2)}$ and use this best classifier to predict on $V_{te}^{(2)}$.

For a given data split, for NoN-GCN-based classification, we train a neural network that consists of two NoN-GCN layers, each followed by dropout layers, followed by a logistic regression classifier (i.e., one fully connected hidden layer). We specifically add this logistic regression classifier on the end of the neural network, rather than directly performing classification from the final NoN-GCN layer, to make the NoN-GCN-based classification as fairly comparable as possible to the regular classification. Note that for synthetic NoNs with two random graph models, we tested a version of the neural network with three NoN-GCN layers. However, because two NoN-GCN layers was as good as three for the majority of the evaluation tests, and because three layers took much more time to compute, we continued with two layers. We also use the ADAM optimizer. We perform a grid search over the following hyperparameters: `learning rate:` $\{0.1, 0.01, 0.001\}$, `dropout:` $\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$, `hidden dimension:` $\{128, 256, 512\}$ and choose the best combination with respect to performance when predicting on $V_{val}^{(2)}$. Then, we use this best classifier to predict on $V_{te}^{(2)}$.

Both DiffPool and our NoN-GCN require initial features. Ideally, they should use the same type of initial features so that they are as fairly comparable as possible. Our NoN-GCN has stricter limitations for what initial features can be used because it requires level 2 nodes' initial features to be in the same low dimensional space as level 1 nodes' initial features, otherwise it does not make sense to aggregate them. So, we determined initial features for our NoN-GCN first. We tested random features of lengths 128, 256, and 512, and nodes' GDVs (each index in the GDV corresponds to the number of times the node participates in that specific graphlet orbit; hence, GDVs are in the same low dimensional space) and found that GDVs were the best. So, we use nodes' GDV as initial features for our NoN-GCN. Thus, we also use nodes' GDVs as initial features into DiffPool.

For synthetic NoNs, we report classification accuracy (# of correct predictions / total # of entities) since class sizes are balanced. For the real-world NoNs, we report area under precision-recall (AUPR), precision@k, recall@k, and F-score@k, since class sizes are not balanced. Here @k refers to the corresponding measure when only considering the top k predictions. That is, for each approach, for each GO term, we rank each protein for which a prediction is made by the probability that it annotated by the given GO term, as determined by the approach's classifier. Then, we compute the corresponding measure on the top k items of the ranked list. To determine k, for each approach, for each GO term, we do the following. We choose the k that maximizes the F-score@k where precision@k is greater than recall@k. We impose precision@k to be greater than recall@k because we believe that in the biomedical domain, precision is more important – fewer but mostly correct predictions (e.g., 9 correct out of 10 made), which corresponds to high precision, is better than a greater number of mostly incorrect predictions (e.g., 300 correct out of 1,000 made), which corresponds to high recall, in terms of potential wet lab validation. By choosing k in this way, we give each classifier the best case advantage. We report precision, recall, and F-score at this k.

We also test if each approach's performance is significantly better than random. That is, given an approach, for each measure, for each GO term, we use a one sample one-tailed t-test (recall that each approach is run 3 times, corresponding to 3 different training/validation/testing splits) to see if the approach's performance is significantly greater than the value expected by random. Then, for each measure, for each

approach, we perform FDR correction over the 131 GO terms. For each measure, for each GO term, any approach with a corrected p-value < 0.05 is considered significantly better than random for that GO term.

S2 Results

S2.1 Synthetic NoNs

We expect an approach only using one level to reach an accuracy of $\frac{\#\text{ of models}}{\#\text{ of labels}}$, i.e., 0.5. To see why, consider the following example using the L1 GDVM approach. Here, there are four possible labels corresponding to the four possible combinations of random graph models at each level: GEO-GEO, GEO-SF, SF-GEO, SF-SF. Since L1 GDVM uses level 1 information, it will be able to distinguish between GEO and SF level 1 networks but not between level 2 nodes with GEO- and SF- topology. So, L1 GDVM will only have enough information to predict $\frac{2}{4} = 0.5$ of the labels correctly.

S2.2 Biological NoN

`http://nd.edu/~cone/NoNs/goinfo.aupr.csv`

Table S1: Raw AUPR scores of the eight relevant approaches for each GO term in each of the six groups.

`http://nd.edu/~cone/NoNs/goinfo.precision.csv`

Table S2: Raw precision scores of the eight relevant approaches for each GO term in each of the six groups.

`http://nd.edu/~cone/NoNs/goinfo.recall.csv`

Table S3: Raw recall scores of the eight relevant approaches for each GO term in each of the six groups.

S2.3 Running times

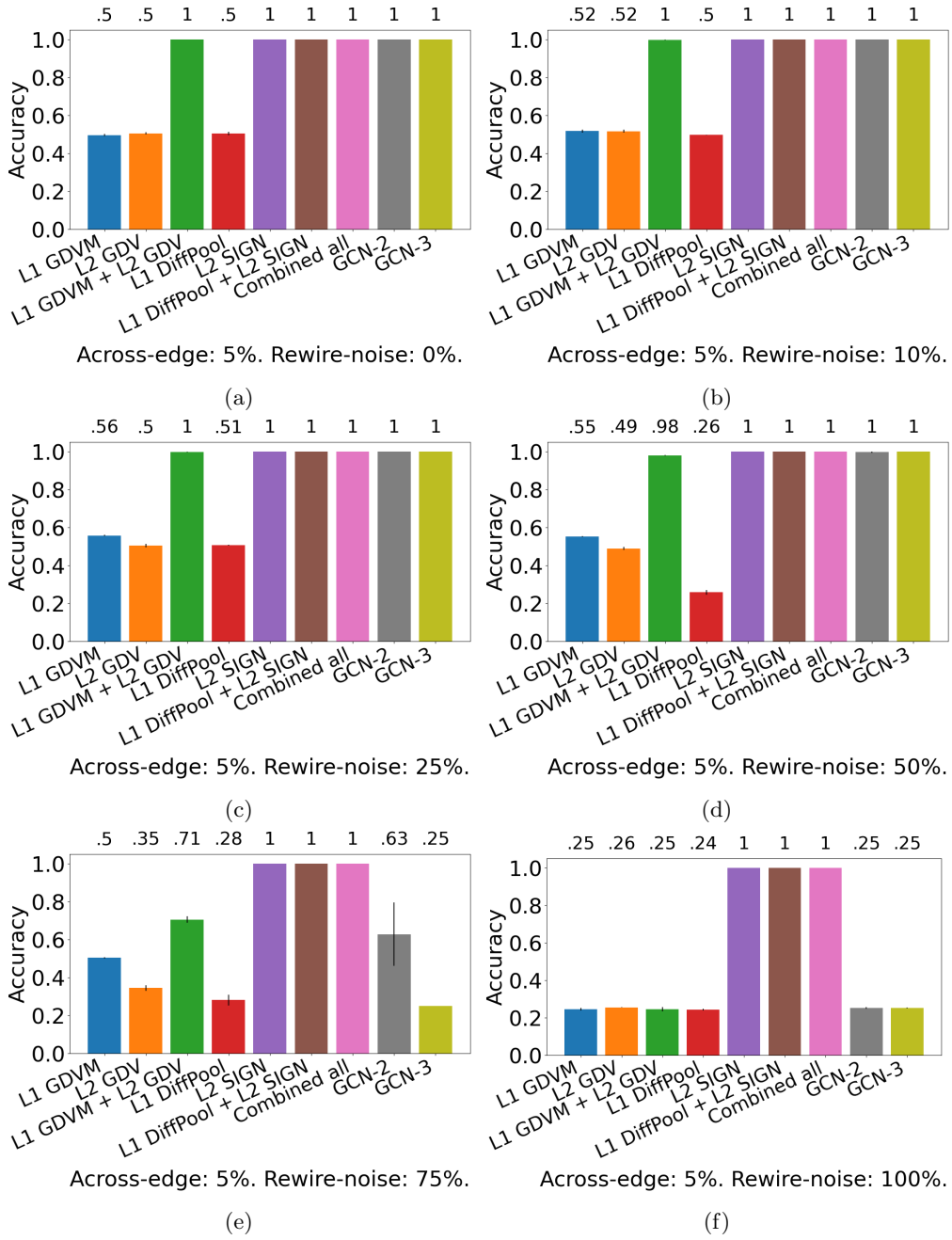
Reported times for all approaches except those involving DiffPool (L1 DiffPool, L1 DiffPool + L2 SIGN, and L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN) are obtained by running on the same machine, fully using one core, for fairness; of course, for practical purposes, some approaches can easily be parallelized given available resources. Training for DiffPool-based approaches must be done on GPU. We report their training times on a cluster machine, which means that their times for training are affected by resource availability/scheduling. While DiffPool-based approaches are not run under the same conditions as other approaches, we still commented on their running times, as in a realistic scenario, approaches may be run using different resources as we have done here.

We run all approaches except those involving DiffPool using one core on a 64-core AMD Opteron 6376 machine. We run approaches involving DiffPool on a cluster machine with Dual Twelve-core 2.2GHz Intel Xeon processors and 4 NVIDIA GeForce GTX 1080 Ti GPUs, accessed through Notre Dame's Center for Research Computing.

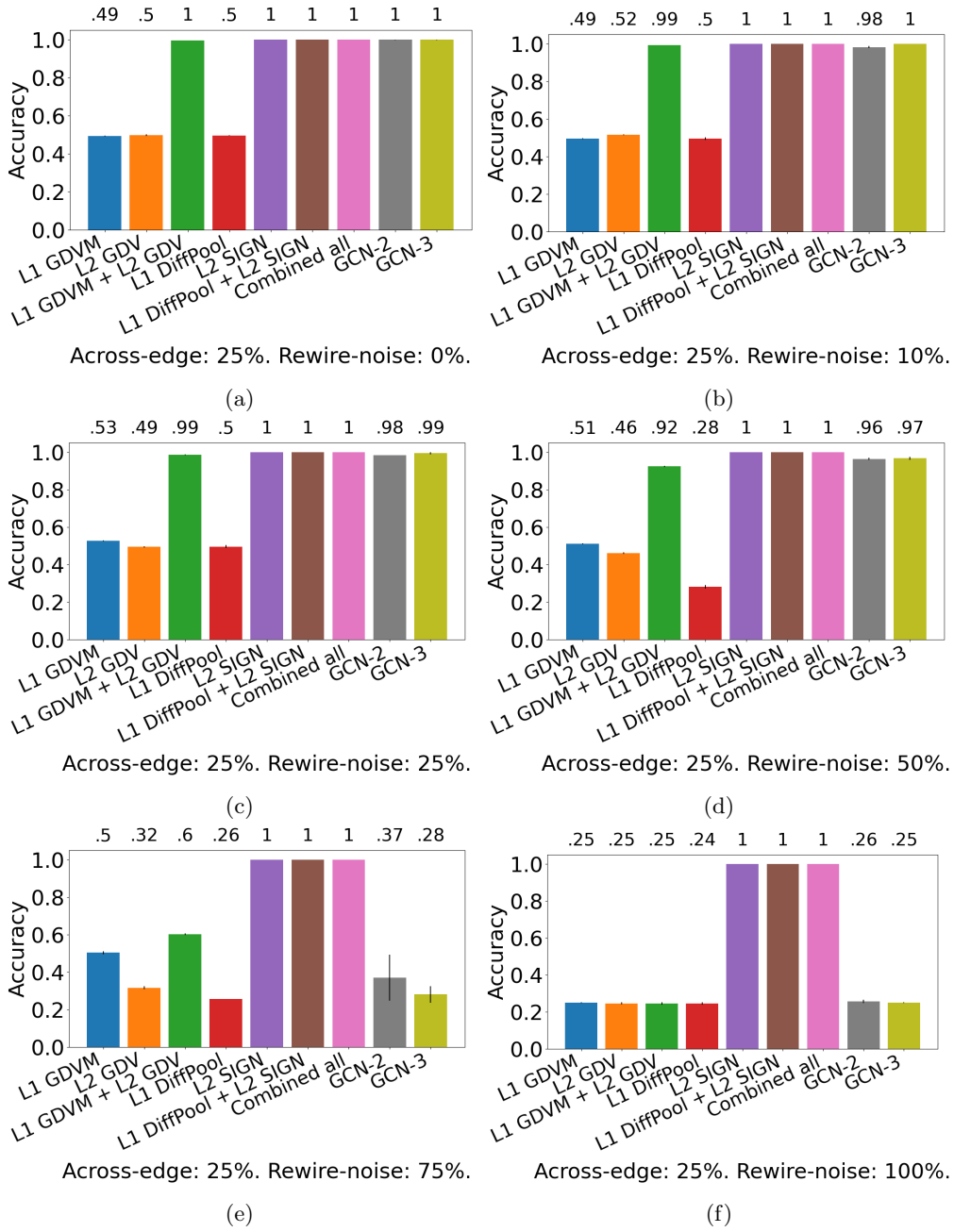
References

[1] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, et al. "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs". In: *Nucleic Acids Research* 25.17 (1997), pp. 3389–3402.

[2] UniProt Consortium. "UniProt: a worldwide hub of protein knowledge". In: *Nucleic Acids Research* 47.D1 (2019), pp. D506–D515.

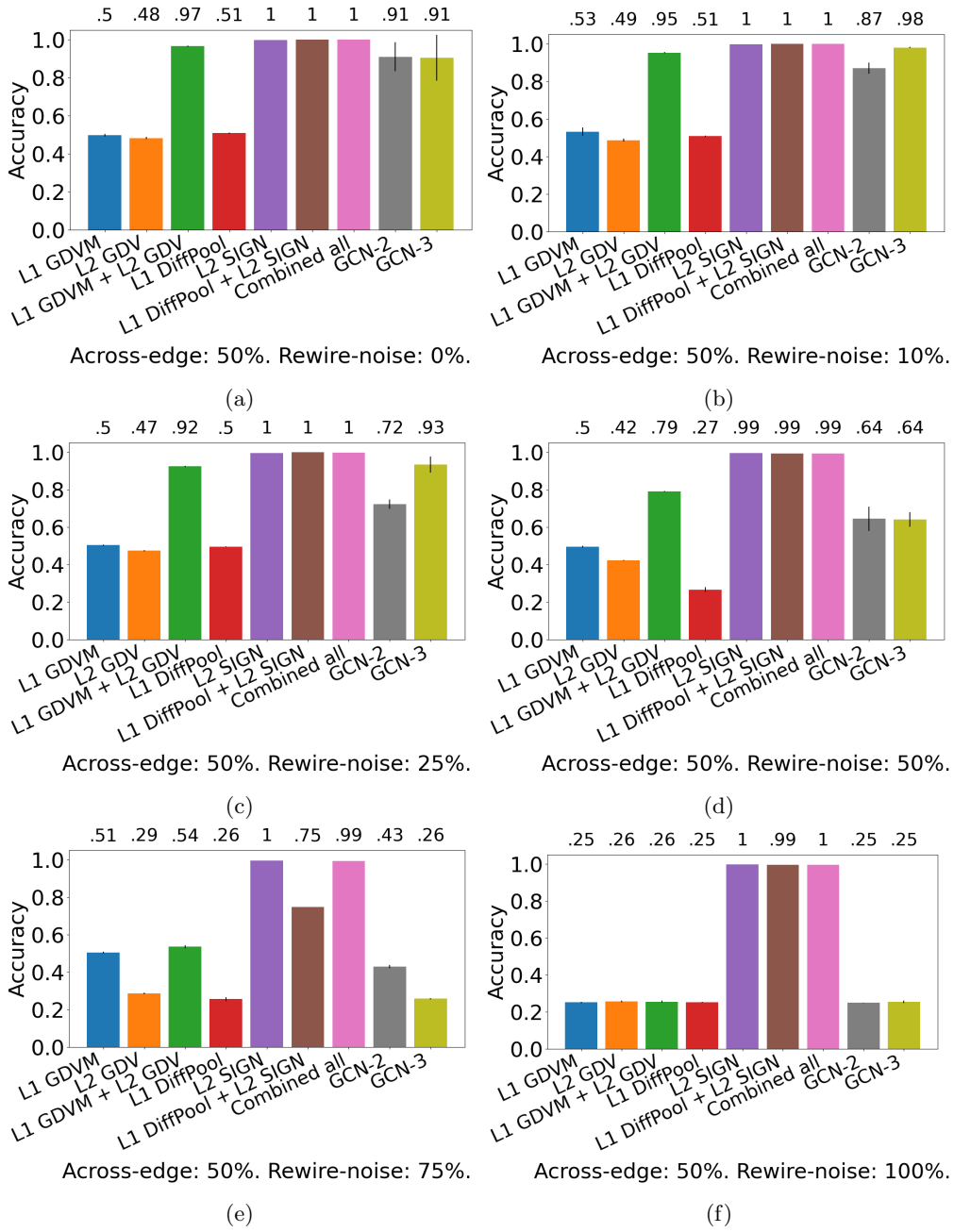


Supplementary Figure S2: Comparison of the nine relevant approaches in the task of entity label prediction for synthetic NoNs with 5% across-edge amount and the following rewire-noise amounts: **(a)** 0%, **(b)** 10%, **(c)** 25%, **(d)** 50%, **(e)** 75%, and **(f)** 100%. "Combined all" refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw prediction accuracies are shown above. "Combined all" refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Accuracy is shown above the bars.

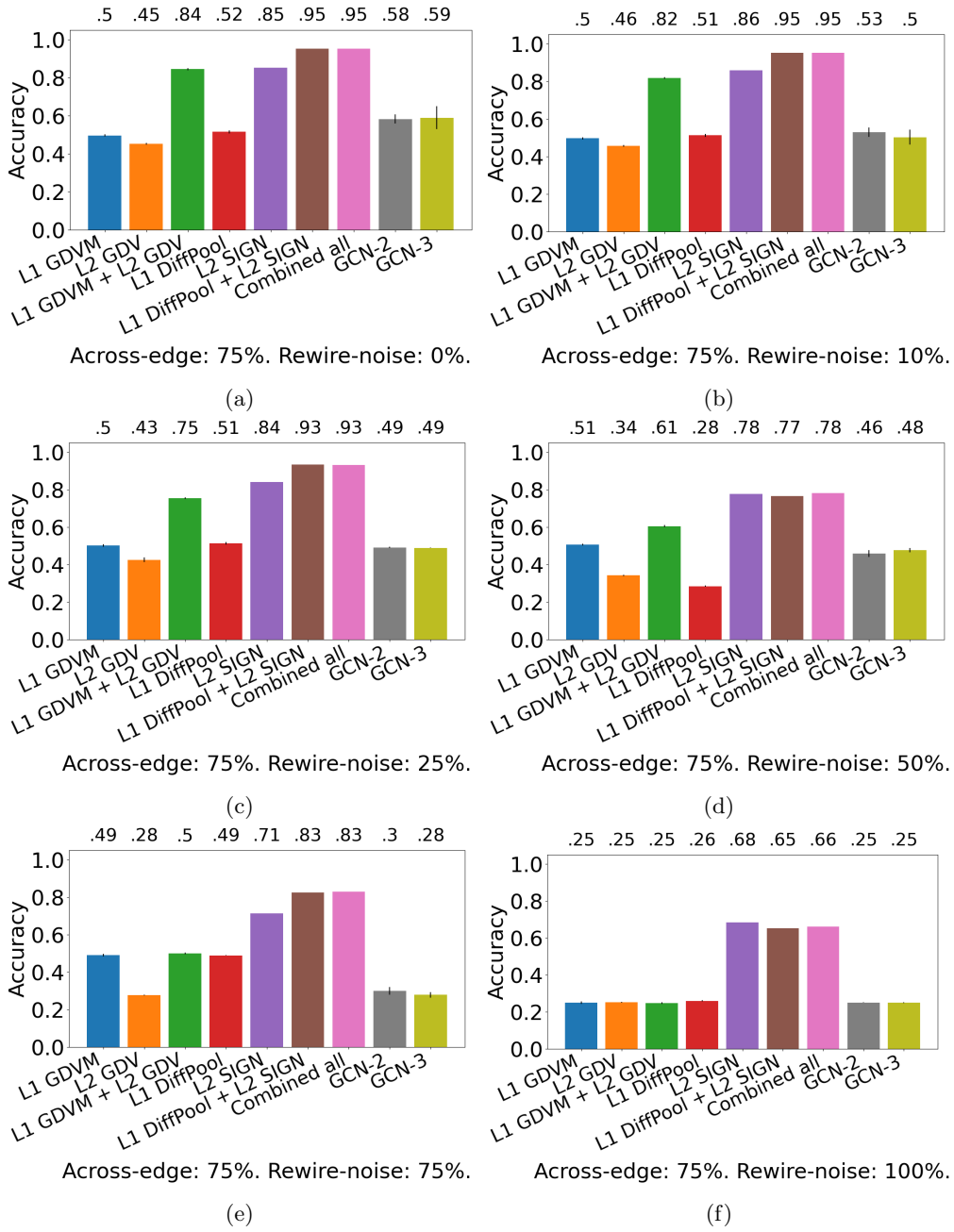


(a) (b) (c) (d) (e) (f)

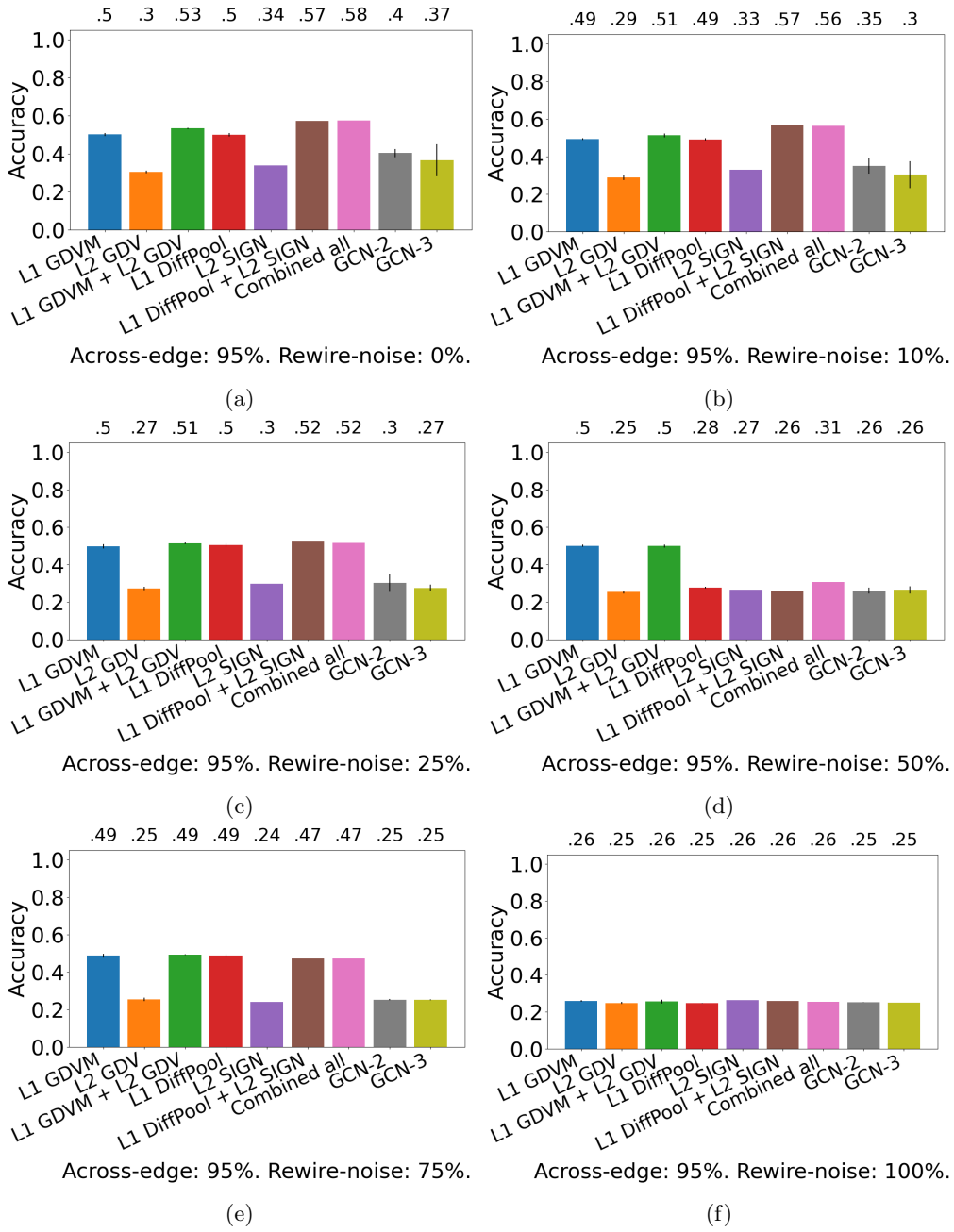
Supplementary Figure S3: Comparison of the nine relevant approaches in the task of entity label prediction for synthetic NoNs with 25% across-edge amount and the following rewire-noise amounts: **(a)** 0%, **(b)** 10%, **(c)** 25%, **(d)** 50%, **(e)** 75%, and **(f)** 100%. "Combined all" refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw prediction accuracies are shown above. "Combined all" refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Accuracy is shown above the bars.



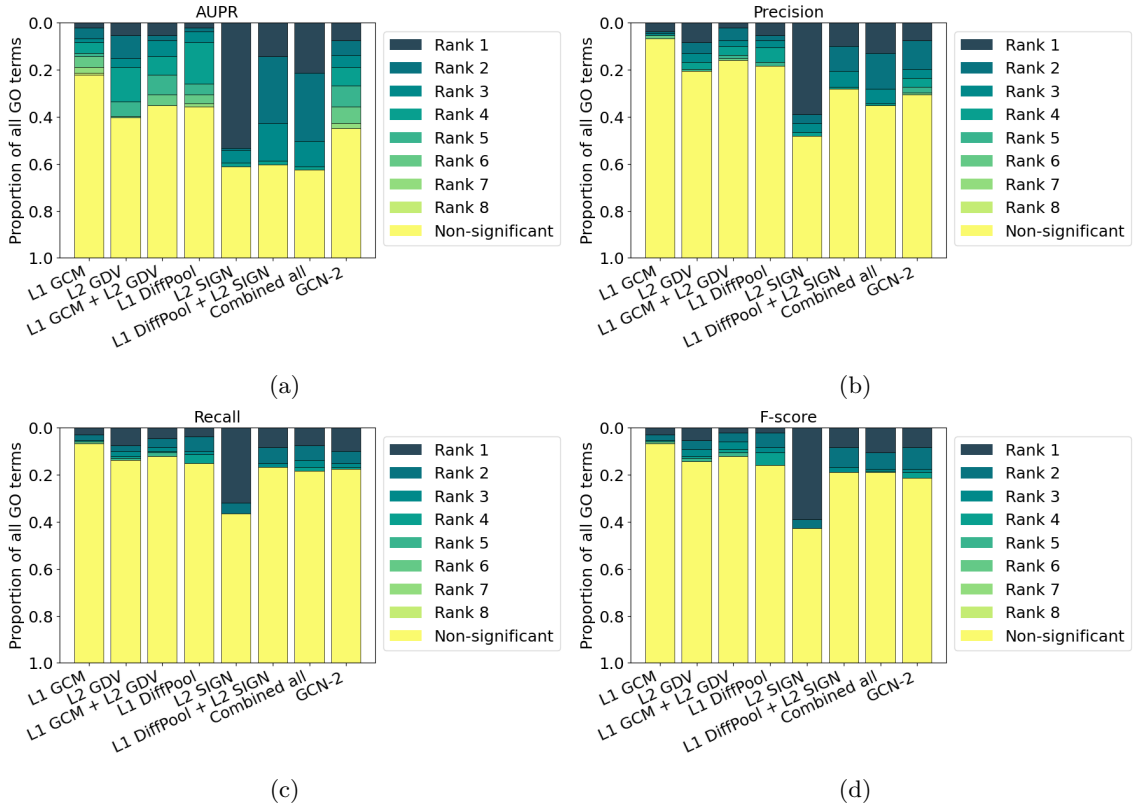
Supplementary Figure S4: Comparison of the nine relevant approaches in the task of entity label prediction for synthetic NoNs with 50% across-edge amount and the following rewire-noise amounts: **(a)** 0%, **(b)** 10%, **(c)** 25%, **(d)** 50%, **(e)** 75%, and **(f)** 100%. "Combined all" refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw prediction accuracies are shown above. "Combined all" refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Accuracy is shown above the bars.



Supplementary Figure S5: Comparison of the nine relevant approaches in the task of entity label prediction for synthetic NoNs with 75% across-edge amount and the following rewire-noise amounts: **(a)** 0%, **(b)** 10%, **(c)** 25%, **(d)** 50%, **(e)** 75%, and **(f)** 100%. "Combined all" refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw prediction accuracies are shown above. "Combined all" refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Accuracy is shown above the bars.



Supplementary Figure S6: Comparison of the nine relevant approaches in the task of entity label prediction for synthetic NoNs with 95% across-edge amount and the following rewire-noise amounts: **(a)** 0%, **(b)** 10%, **(c)** 25%, **(d)** 50%, **(e)** 75%, and **(f)** 100%. "Combined all" refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw prediction accuracies are shown above. "Combined all" refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Accuracy is shown above the bars.



Supplementary Figure S7: Summarized results of the eight relevant approaches in the task of protein functional prediction for evaluation measures **(a)** AUPR, **(b)** precision, **(a)** recall, and **(a)** F-score. For each GO term (out of the 131 total), we rank the eight approaches' classification performances from best (rank 1) to worst (rank 8). If an approach's performance is not significantly better than expected by random we deem it "non-significant" instead. Then for each approach, we calculate the proportion of times it achieves each rank. "Combined all" refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN.

`http://nd.edu/~cone/NoNs/goinfo.f-score.csv`

Table S4: Raw F-scores of the eight relevant approaches for each GO term in each of the six groups.

	Feature extraction	Training (1 epoch)	Total
L1 GDVM	485.0	2.2	487.2
L2 GDV	2.1	1.5	3.6
L1 GDVM + L2 GDV	487.1	2.1	489.2
L1 DiffPool	485.0	140.1	625.3
L2 SIGN	6.4	17.6	23.4
L1 DiffPool + L2 SIGN	491.4	25.6	516.4
Combined all	493.5	29.4	522.5
GCN-2	487.1	30.3	517.1
GCN-3	487.1	128.8	615.1

Table S5: Running times of each approach in seconds. "Combined all" refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN.

[3] Daniele Grattarola and Cesare Alippi. "Graph neural networks in tensorflow and keras with spektral". In: *arXiv preprint arXiv:2006.12138* (2020).

[4] Aditya Grover and Jure Leskovec. "node2vec: Scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 855–864.

[5] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

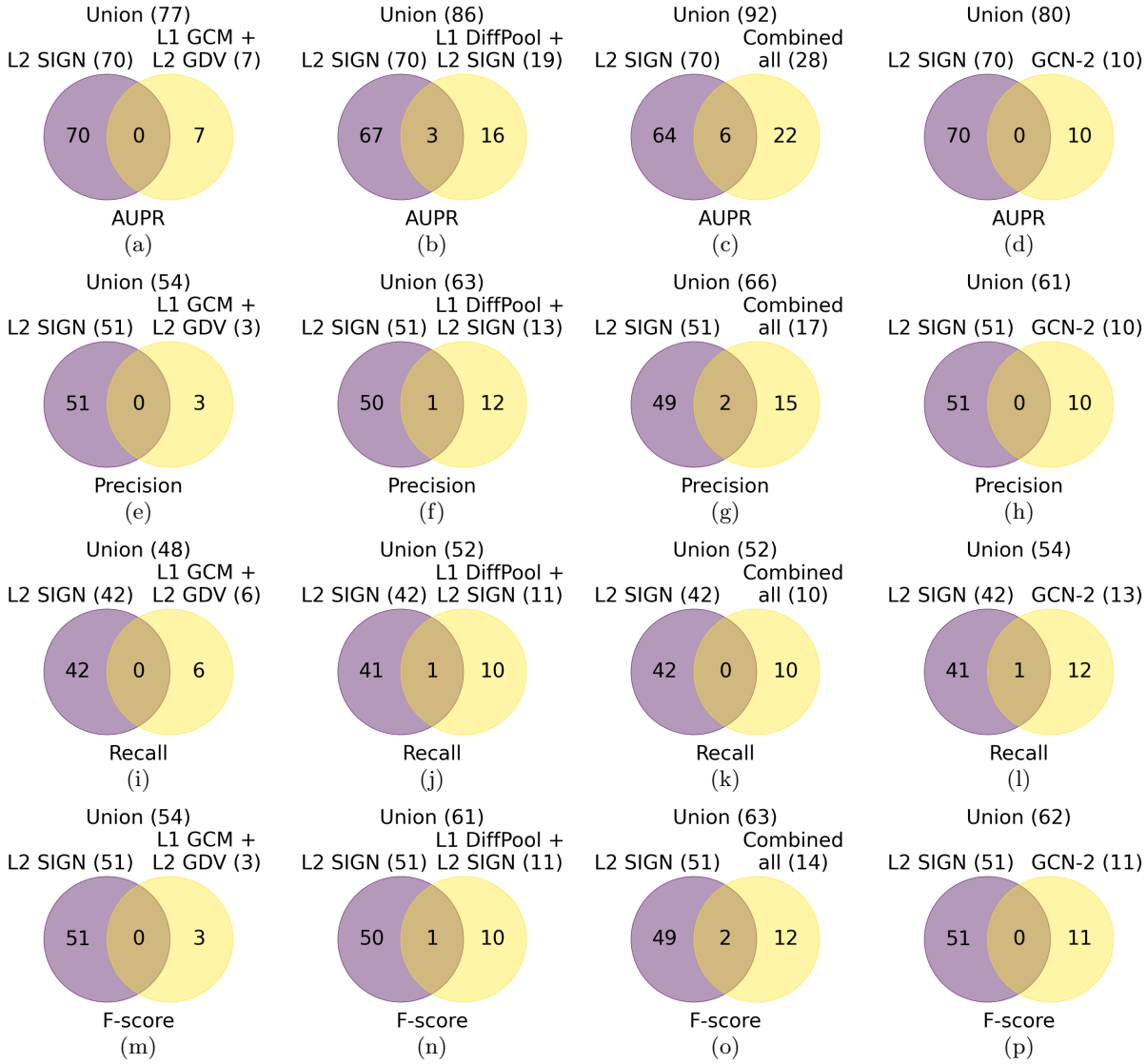
[6] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. "Diffusion improves graph learning". In: *33rd Conference on Neural Information Processing Systems*. 2019.

[7] Federico Monti, Karl Otness, and Michael M Bronstein. "Motifnet: a motif-based graph convolutional network for directed graphs". In: *2018 IEEE Data Science Workshop (DSW)*. IEEE. 2018, pp. 225–228.

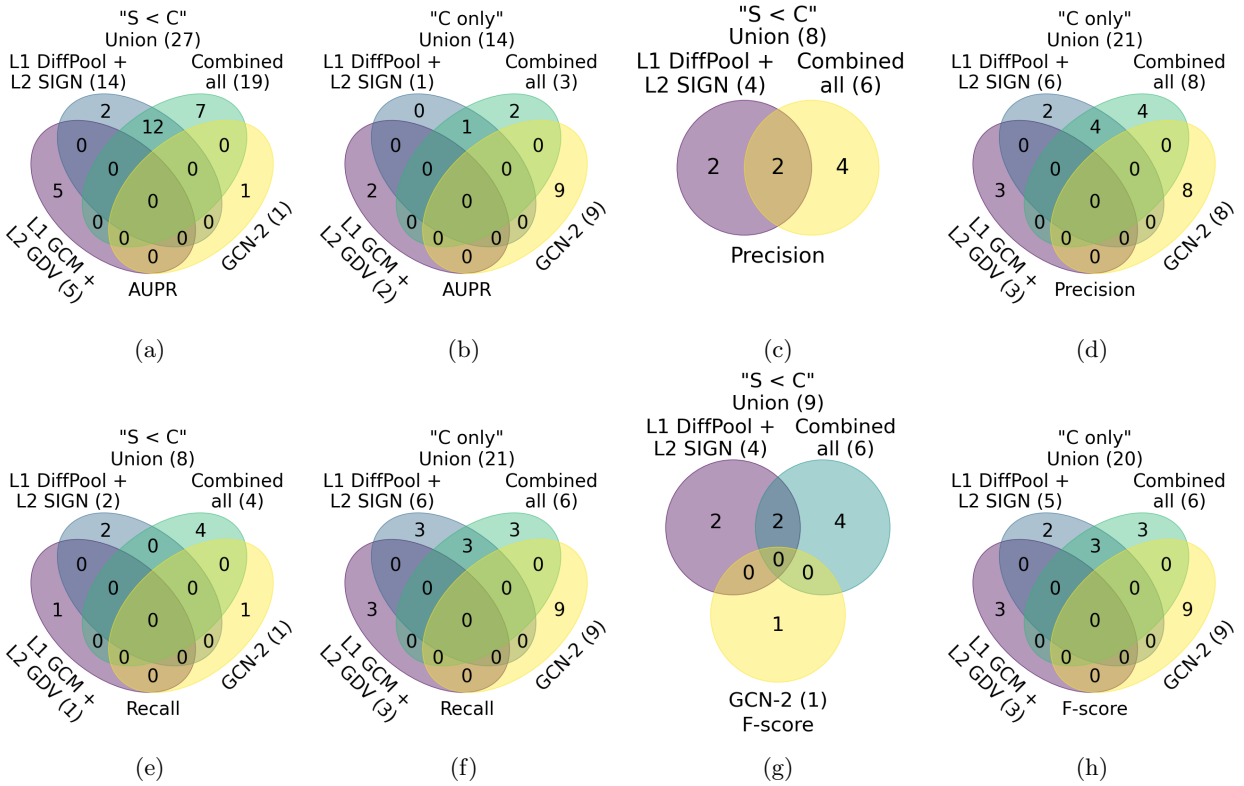
[8] PDB. *PDB101: Learn: Guide to Understanding PDB Data: Resolution*. 2020. URL: `http://pdb101.rcsb.org/learn/guide-to-understanding-pdb-data/resolution`.

[9] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, et al. "BioGRID: a general repository for interaction datasets". In: *Nucleic Acids Research* 34.suppl_1 (2006), pp. D535–D539.

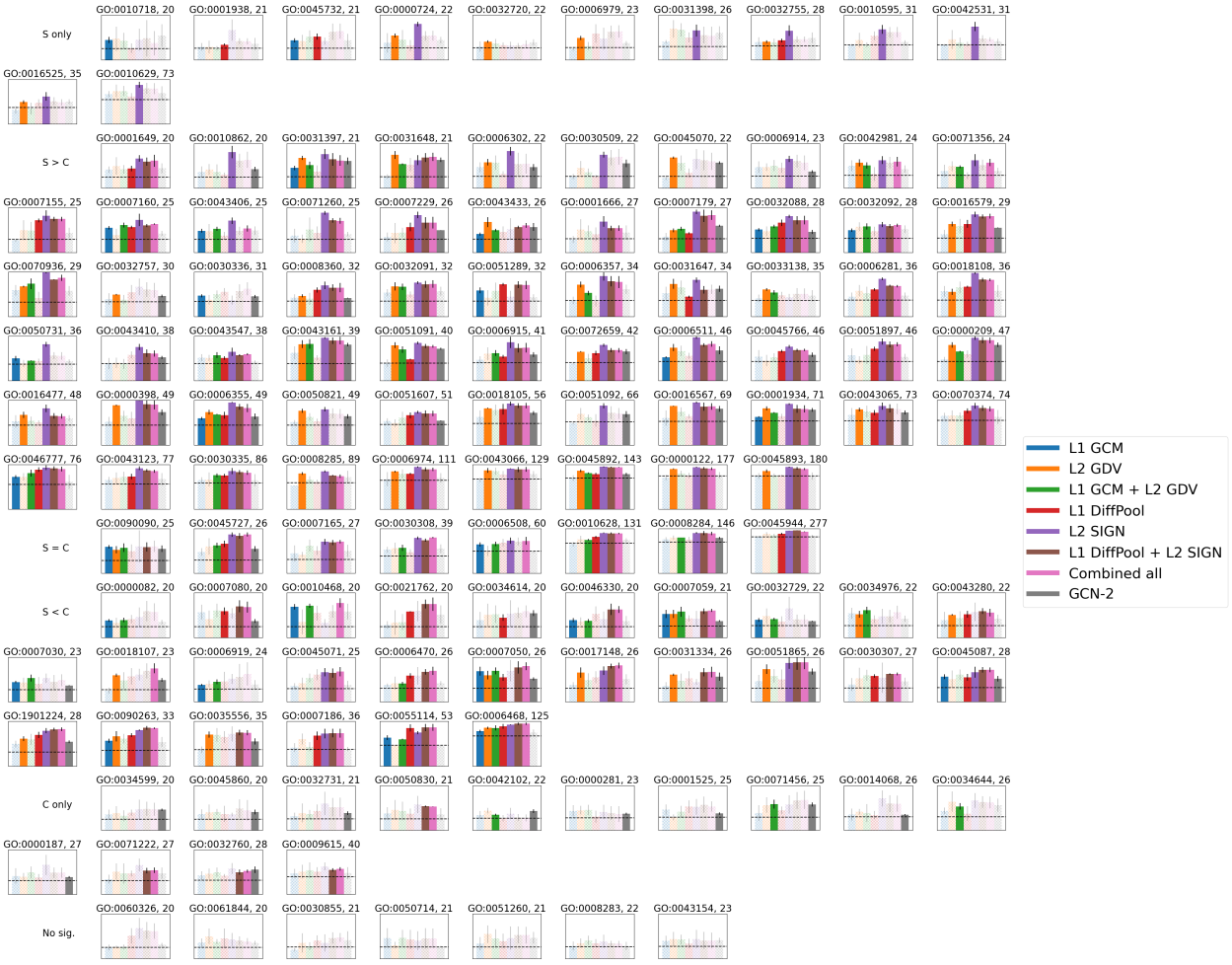
[10] Ömer Nebil Yaveroğlu, Noël Malod-Dognin, Darren Davis, et al. "Revealing the hidden language of complex networks". In: *Scientific Reports* 4 (2014), p. 4547.



Supplementary Figure S8: Overlap of GO terms for which L2 SIGN is the best with those for which **(a, e, i, m)** L1 GCM + L2 GDV, **(b, f, j, n)** L1 DiffPool + L2 SIGN, **(c, g, k, o)** Combined all (aka L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN), and **(d, h, l, p)** GCN-2 are the best in terms of **(a, b, c, d)** AUPR, **(e, f, g, h)** precision, **(i, j, k, l)** recall, and **(m, n, o, p)** F-score.



Supplementary Figure S9: Overlaps of the four combined level approaches for groups **(a, c, e, g)** "S < C" and **(b, d, f, h)** "C only" in terms of **(a, b)** AUPR, **(c, d)** precision, **(e, f)** recall, **(g, h)** F-score.



Supplementary Figure S10: Classification performance of the eight relevant approaches for each GO term in terms of AUPR. GO term IDs and the number of positive instances for that GO term are shown above. Random performance is indicated by the dotted black line. Approaches with performance not significantly greater than random are shown in a lighter shade. GO terms are split into the six groups based on how single versus combined level approaches perform. "Combined all" refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw scores for each approach for each GO term can be found in Supplementary Table S1.

Supplementary Figure S11: Classification performance of the eight relevant approaches for each GO term in terms of precision. GO term IDs and the number of positive instances for that GO term are shown above. Random performance is indicated by the dotted black line. Approaches with performance not significantly greater than random are shown in a lighter shade. GO terms are split into the six groups based on how single versus combined level approaches perform. "Combined all" refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw scores for each approach for each GO term can be found in Supplementary Table S2.

Supplementary Figure S12: Classification performance of the eight relevant approaches for each GO term in terms of recall. GO term IDs and the number of positive instances for that GO term are shown above. Random performance is indicated by the dotted black line. Approaches with performance not significantly greater than random are shown in a lighter shade. GO terms are split into the six groups based on how single versus combined level approaches perform. "Combined all" refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw scores for each approach for each GO term can be found in Supplementary Table S3.

Supplementary Figure S13: Classification performance of the eight relevant approaches for each GO term in terms of F-score. GO term IDs and the number of positive instances for that GO term are shown above. Random performance is indicated by the dotted black line. Approaches with performance not significantly greater than random are shown in a lighter shade. GO terms are split into the six groups based on how single versus combined level approaches perform. "Combined all" refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw scores for each approach for each GO term can be found in Supplementary Table S4.