Majority Vote Cascading: A Semi-Supervised Framework for Improving Protein Function Prediction

John Lazarsfeld*
john.lazarsfeld@tufts.edu
Tufts University

Jonathan Rodríguez*
jonathan.rodriguez@tufts.edu
Tufts University

Mert Erden
mert.erden@tufts.edu
Tufts University

Yuelin Liu
yuelin.liu@tufts.edu
Tufts University

Lenore J. Cowen†
lenore.cowen@tufts.edu
Tufts University

ABSTRACT

A method to improve protein function prediction for sparsely annotated PPI networks is introduced. The method extends the DSD majority vote algorithm introduced by Cao et al. to give confidence scores on predicted labels and to use predictions of high confidence to predict the labels of other nodes in subsequent rounds. We call this a *majority vote cascade*. Several cascade variants are tested in a stringent cross-validation experiment on PPI networks from *S. cerevisiae* and *D. melanogaster*, and we show that for many different settings with several alternative confidence functions, cascading improves the accuracy of the predictions. A list of the most confident new label predictions in the two networks is also reported.

Code, networks for the cross-validation experiments, and supplementary figures and tables appear at http://bcb.cs.tufts.edu/cascade.

KEYWORDS

PPI networks; protein function prediction; graph diffusion; semi-supervised learning

ACM Reference Format:
John Lazarsfeld, Jonathan Rodríguez, Mert Erden, Yuelin Liu, and Lenore J. Cowen. 2019. Majority Vote Cascading: A Semi-Supervised Framework for Improving Protein Function Prediction. In *10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (ACM-BCB '19), September 7–10, 2019, Niagara Falls, NY, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3307339.3342135

1 INTRODUCTION

Functional label prediction is one of the best-known and most well-studied problems on protein-protein interaction networks [6, 10, 11, 13, 16, 18, 19, 22]. Many prediction methods take advantage of what social scientists call "homophily," [15] which means that nodes tend to be located in proximity to other, similar nodes. In the context of functional label prediction, knowing the set of functions in a node's local neighborhood can be used to help correctly predict its own label.

Consider the simplest method for exploiting homophily to predict function, the majority vote algorithm, which was introduced by [18] in 2000. In the original version of this algorithm, all direct neighbors with known annotations vote for its labels, and the label receiving the most votes is assigned to the node of unknown function. Our starting point for this paper is to observe that most implementations of this algorithm discard information when they output their predictions. In particular, a scenario where all neighbors of a node vote for the same label is very different from one in which there are many different labels among a node's neighbors, and each label gets nearly an equal number of votes (See Figure 1). In the first scenario (see Figure 1a), we should be more confident that our prediction is correct than in the second scenario (Figure 1b), and yet this extra confidence information is typically discarded, rather than preserved where it could inform downstream analysis.

We are interested, therefore, in generalizing voting-based function prediction schemes to explicitly output node confidences in their predictions. The set of majority vote algorithms we consider here do not use the simple direct-neighbor voting described above, but rather use variants of asking the k-closest nodes in cDSD distance [2] to vote on their functional labels. It was shown in [2]

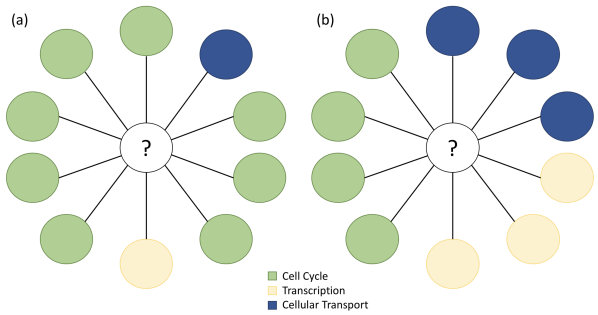

Figure 1: Both center nodes in Figures 1a and 1b will be labeled cell cycle (green) by a simple majority vote algorithm that has all direct neighbors vote for their label. However, in Figure 1a, the winning label wins by a much higher margin than in Figure 1b, so we should have higher confidence in the winning label.

*Both authors contributed equally to this research.
†To whom correspondence should be addressed: lenore.cowen@tufts.edu

ACM-BCB '19, September 7–10, 2019, Niagara Falls, NY, USA

ACM ISBN 978-1-4503-6666-3/19/09…$15.00
https://doi.org/10.1145/3307339.3342135

that this variant of majority voting produced much superior performance on functional labeling tasks in the *S. cerevisiae* PPI network as compared to many other classical function prediction algorithms.

In this paper, we propose three different ways of assigning confidences in functional labeling voting schemes, and we apply our methods to a new function prediction scheme that assigns tentative function predictions in rounds, where high confidence function predictions are assumed to be correct and thus contribute to votes in subsequent rounds. We call this *majority vote cascading*. We investigate setting the parameters for the number of rounds and thresholds for prediction confidence in order to improve function prediction, showing that our majority vote cascades improve the performance of multiple variants of cDSD-based majority vote methods on the functional labeling tasks based on the *S. cerevisiae* (yeast) PPI network.

We are particularly interested in using a method such as majority vote cascading on networks that are less well annotated than the yeast network. We therefore also investigate how these methods perform on the PPI network of *D. melanogaster* (fly). We note that only 43% of nodes in the fly network have at least one functional label at depth four or below from the hierarchy roots of the Biological Process or Molecular Function categories of the Gene Ontology compared to 83% of nodes in the yeast network. We find similar trends for cascading in the fly network as in yeast and report for both a list of our most confident new predictions.

The structure of the remainder of this paper is as follows: we first review the definitions of DSD and cDSD from [2, 3] and discuss the variants of cDSD majority vote that we investigate. Section 2 describes our data and cascading framework in more detail as well as introduces our new confidence scoring functions. Section 2.3 describes our cross validation evaluation framework. In Section 3, we present our results on the yeast and fly networks and discuss issues and open problems in Section 4.

1.1 Review of DSD and cDSD metrics

In a series of two papers [2, 3], Cao et al. introduced a novel distance metric called "Diffusion State Distance" (DSD) and "Confidence-based Diffusion State Distance" (cDSD) that they argued could better capture the dissimilarity between nodes in a PPI network.

DSD comes from a variant of Network Propagation [7] that finds two nodes connected by multiple short paths in a network to be more similar than nodes that are only connected by a single path. Furthermore, DSD downweights paths that go through hub nodes, which are less functionally informative. cDSD generalizes DSD to take into account edge confidences in a network in the natural way. The formal definition of cDSD appears in section 2.1.5.

Cao et al. [2, 3] looked at different classical methods for predicting protein function, including ordinary majority vote, functional flow, and multiway cut, and they showed that a simple method that performed majority vote on the k-nearest neighbors in cDSD distance performed best. We look at several variants of cDSD majority voting and add our new cascading framework to each.

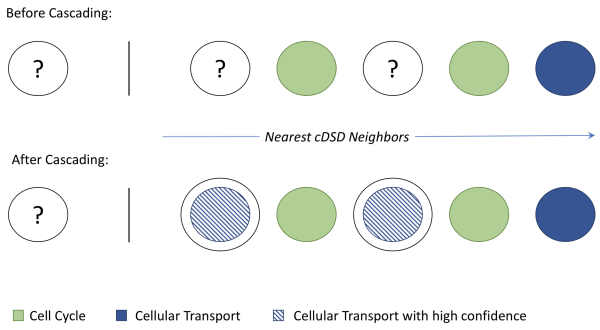

Figure 2: To predict the function of the left-most node, we consider its nearest cDSD neighbors in a majority vote. Before cascading (above), the unknown labels of the first and third closest neighbors mean they cannot contribute any information in making a prediction. By cascading (below), we hope that neighbors with unknown function might receive *pseudo-labels* with high confidence, which allows them to contribute to the predictions of other nodes.

1.2 Confidences and cascading

While predicting the functional label of a node given the labels of its closest cDSD neighbors is effective, networks with sparse annotations are disadvantaged by close cDSD neighbors with unknown labels that cannot contribute to majority votes. Consider Figure 2, where to predict the function of the left-most node we consider the function of its closest cDSD neighbors in a majority vote. In the top portion of the figure (before cascading), the outcome of a majority vote and thus the resulting prediction is limited by the unknown labels of the first and third closest cDSD neighbors. The goal of cascading is to increase the likelihood of the bottom portion of the figure (after cascading). There, we can imagine that those initially-unlabeled neighbors, in their own majority votes, were assigned predicted functional labels with high confidence. Now in later rounds, these highly-confident predictions are treated as *pseudo-labels* [23], which allow them to vote with weight proportional to their confidences in the majority voting of other unknown nodes. Augmenting training with high-confidence pseudo-labels is a standard approach in semi-supervised learning frameworks [4, 14, 23]. As the figure shows, the information that high confident predictions propagate can update the predicted label of other nodes.

Given that cascading allows majority vote algorithms to treat the pseudo-labels assigned to high confidence predictions as truth, the effectiveness of cascading is dependent on the quality of the confidence functions. The three confidence functions designed in this paper (and introduced in detail later in Section 2.2.3) all consider the label distribution from a majority vote in order to assign a confidence score.

2 METHODS

2.1 Data

The best type of PPI or Protein-Protein association network to use for functional label prediction is currently a source of vigorous debate, where [12] did an extensive recent study on what might be

the "best" source of interaction data for human networks to predict genes associated with human disease. Based on the conclusion of their article, it seems that when looking at the tradeoff between the number of reported interactions and the quality and curation standards for the reported interactions, the two extremes (small number of high-quality edges or many edges, regardless of quality) did better than intermediate networks. However, this study was carried out on *unweighted* networks: presumably, with weighted networks the edge confidence function could be tuned to adjust for the quality of the predictions, and more edges are better. We decided to extract our PPI network from BioGRID to better mimic the experiments in [3]. However, we would expect the trends in performance that we see using the different methods would hold in the more permissive and extensive protein-protein association networks as well, such as those contained in the STRING database [21].

2.1.1 Yeast Physical Protein Interaction Network from BioGRID. A *Saccharomyces cerevisiae* protein-protein physical interaction network is constructed as follows: the list of 5,176 verified ORFs downloaded from the *Saccharomyces* Genome Database (version date April 2, 2019) defines the set of nodes, and the 167,800 protein-protein physical interactions from BioGRID (version 3.5.170, downloaded March 20, 2019) [20] between nodes that are verified by at least one wet-lab experiment define the set of edges. We remove edge redundancy, self-loops, and edges incident to unverified ORF nodes, and we extract the largest connected component to obtain an undirected graph with $n = 5{,}143$ nodes (denoted by $V = \{v_1, ..., v_n\}$) and $m = 103{,}550$ unique, undirected edges (denoted by E). Additionally, we assign confidence to PPIs using a scoring scheme identical to the method found in [2]. Each PPI is thus assigned a confidence score that translates to an edge weight. We denote $W = \{w_{ij}\}_{ij}$ as the weight matrix, where $0 < w_{ij} < 1, \forall (v_i, v_j) \in E$, and $w_{ij} = 0, \forall (v_i, v_j) \notin E$. We denote this graph resulting from BioGRID PPI network as $G(V, E, W)$.

2.1.2 Fly Physical Protein Interaction Network from BioGRID. A *Drosphilia melanogaster* protein-protein physical interaction network is similarly constructed by considering the physical interactions from BioGRID (version 3.5.170), and again removing edge redundancy and self-loops. We use the the same interaction confidence scoring scheme described above to assign weights to edges, and the result is a graph with 8,905 nodes with 52,223 unique, undirected edges.

2.1.3 MIPS Functional Categories. For *S. cerevisiae*, the Munich Information Center for Protein Sequences (MIPS) functional catalogue (FunCat, version 2.1 [17]), gives a leveled hierarchy of functional labels. Similar to [2, 3], we present results for MIPS annotations at the first level (4,386 nodes with 10,476 annotations in 17 functional categories in *S. cerevisiae*, second level (4,372 nodes with 12,266 annotations in 74 functional categories), and third level (4,025 nodes with 9,371 annotations in 154 functional categories), where the number of categories, and hence the difficulty of the classification task, increases for the higher levels of MIPS. We denote these three sets of labels as MIPS1, MIPS2, and MIPS3 in our results further below.

2.1.4 GO Functional Categories. For both *S. cerevisiae* and *D. melanogaster*, we also present results using annotations from the more popular Gene Ontology (GO) [1]. We consider the union set of labels from the molecular function and biological process categories of the GO hierarchy (OBO format version 1.2, downloaded on 2019-03-19). We download associations from FuncAssociate 3.0 (downloaded on 2019-03-22), propagating terms so that ancestors inherit all the terms of their children.

In particular, we include annotations supported by experimental and high throughput evidence codes (EXP, IMP, HMP, HEP, IDA, IGI, HGI, IPI, IEP, and HDA). Using the same criteria from [3] and suggested by [9], we define the set of *informative* GO terms as those that 1) are at least three levels below the root and 2) annotate more than 50 proteins in our dataset. For our GO experiments in this paper, to better match our MIPS experiments, we restrict our attention to the informative GO terms that are *exactly* four levels down from either root (Biological Process or Molecular Function). This mimics a single level of the MIPS hierarchy. For the yeast network, the result is 19 informative molecular process terms and 140 informative biological process terms, where 4,345 of 5,143 nodes receive at least one annotation (31,294 total annotations). For the fly network, the result is 12 informative molecular process terms and 168 informative biological process terms, where 3,866 of 8,905 nodes receive at least one annotation (30,928 total annotations). Note that this is equivalent to 43% of nodes in the fly network having at least one label, while 84% of nodes in the yeast network have at least one label. We denote this label category as GO-MFBP-4 in our results below.

2.1.5 Computing the cDSD Similarity Matrix. The matrix of cDSD distances between all nodes in the graph is computed using the software provided by [2]. The cDSD distance between nodes u and v is calculated as follows: first, given our graph $G(V, E, W)$, we denote $P = \{p_{ij}\}_{i,j=0}^{n}$ as the n-dimensional one-step transition matrix, where the (i, j)'th entry is given by:

$$p_{ij} = \begin{cases} \frac{w_{ij}}{\sum_{l=1}^{n} w_{il}} & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

Here, P represents the probability of reaching each neighbor in a one-step random walk, where neighbors corresponding to higher edge weights are reached with higher probability. The t-step transition probability matrix is then defined as $P^{\{t\}} = P^t$ for all positive t. We denote $He^{\{t\}}(v_i, v_j)$ as the expected number of times a random walk of t-steps starting at node v_i will visit node v_j, which can be calculated by $\sum_{l=0}^{t} p_{ij}^{\{l\}}$. The n-dimensional vector $He^{\{t\}}(v_i), \forall v_i \in V$ can be constructed accordingly. Finally, for a fixed number of steps t in the random walk, the cDSD distance between nodes u and v is given by:

$$\text{cDSD}^{\{t\}}(u, v) = ||He^{\{t\}}(u) - He^{\{t\}}(v)||_1$$

where we set $t = 7$ as recommended by [3] and [2].

2.2 Cascading Framework

2.2.1 Overview. We now more formally describe the procedure that converts a function prediction method (with or without confidences on label predictions) to a procedure that assigns function predictions and associated confidences in those predictions in an iterative series of steps that we call a *cascade*.

The input is a graph $G = (V, E, W)$ that is partially labeled, where each labeled node also has an associated label confidence score between 0 and 1 inclusive. Denote V_k as the set of labeled nodes, where each label is either a known annotation or a highly-confident predicted annotation (which we assign as a pseudo-label) from a previous round in the cascade. A node with a known annotation is assigned a label confidence score of 1, while a node with a pseudo-label is given a label confidence score equivalent to its prediction confidence value from the previous round in the cascade. The remaining, unlabeled nodes are denoted by the set V_u.

The cascading procedure then takes as input four other main parameters:

i. R - a number of cascade rounds.
ii. h - a 'high-confidence' percentile threshold.
iii. F - an algorithm that generates a predicted label (denoted $v.pred$) for every node $v \in V_u$.
iv. C - a function that, given the set of predictions generated by F, assigns a prediction confidence score between 0 and 1 (denoted $v.conf$) for every node $v \in V_u$.

The final output is the same graph G, where every node has a label and a label confidence score. We primarily tested our framework using the underlying function prediction methods of [3] and [2], which are based on several variants of k-nearest-cDSD-neighbor voting schemes (see Section 2.2.2). These methods were shown in [3] and [2] to outperform many of the classical methods for protein functional label prediction.

During each cascade round, we use F and the information on nodes in V_k to make a label prediction for every node in V_u. Then using the prediction confidence function C and given the high-confidence threshold h (we describe the confidence functions in more detail in section 2.3.3), we assign pseudo-labels to the nodes most confident in their predictions, and we also assign label confidence scores to these nodes with values equal to their prediction confidence scores generated by C. Subsequently, these highly-confident nodes are moved from V_u to V_k for the next round of the cascade. As we progress through additional rounds, we expect the size of V_u to decrease as more pseudo-labels are assigned.

Thus for $0 \leq r < R$, the cascading procedure operates as follows:

(1) For all $v \in V_u$, call F to generate $v.pred$ and call C to generate $v.conf$.
(2) Let $T_r \subseteq V_u$ denote the set of nodes with 'high-confidence' predictions (where $v.conf$ is in the top h percentile of all confidence scores) at round r.
(3) For each $v \in T_r$ assign $v.pred$ as a 'pseudo-label' of v, and assign $v.conf$ as the label confidence score of v. Remove v from V_u and add v to V_k.
(4) If $V_u \neq \emptyset$ and $r < R$, repeat the procedure for round $r + 1$.

During the initial round of the cascading procedure, every node initially in V_u is assigned a predicted label. Nodes with low confidence predictions (based on C and h) are also assigned new predictions in subsequent rounds, which may change based on the influence of the nodes assigned pseudo-labels (which are treated as known labels by the prediction algorithms).

As described in step (4), the cascading procedure will continue for all R rounds unless T_r becomes empty.

2.2.2 Protein Function Prediction Methods. For our underlying prediction methods, we use four variations of neighborhood majority voting algorithms to predict the function of unlabeled proteins: the first two variants, cDSD-MV and cDSD-WMV, were introduced by [2, 3]. In these variants, only nodes among the k-nearest neighbors in DSD distance get to vote. The final two variants are newly defined in this paper in the spirit of sparser annotation: additional further neighbors are invited to participate in the vote until the total number of nodes with known labels that get to vote reaches k. As we shall see below, cascading improves the first two variants usually by a greater margin than the more flexibly defined neighborhood voting of the second two variants. More formally, define:

- **(cDSD-MV)** - cDSD-Majority vote: the k-nearest neighbors of node u under the cDSD metric vote for each of their known (or pseudo) labels, and each neighbor votes proportionally to their label confidence score. If a neighbor has no known labels, it does not contribute any votes. The label l receiving the most votes is assigned as the predicted label of u.
- **(cDSD-WMV)** - cDSD-Weighted Majority Vote: identical to cDSD-MV, but each neighbor of u votes for its labels with a voting power proportional to the reciprocal of its cDSD distance to u and also to its label confidence score.
- (**cDSD-MV-known**) - cDSD-Majority Vote with "known" neighbors: here, we expand the local neighborhood of u until there are k neighbors under the cDSD metric that have a known label or pseudo-label. In this regard, every neighbor contributes a vote.
- (**cDSD-WMV-known**) - cDSD-Weighted Majority Vote with "known" neighbors: similar to cDSD-MV-known, but now using a weighted majority vote, as in cDSD-WMV.

In all 4 variants, when predicting the label of node u, if multiple labels received the most number of votes, the predicted label is assigned by a tiebreak according to lexicographical order. As recommended by [3] and [2], we set $k = 10$ for all instances of cDSD-MV, cDSD-WMV, cDSD-MV-known, and cDSD-WMV-known. Similar to the results of [3], we find that small increases or decreases in the value of k do not substantially impact performance (see supplementary tables).

2.2.3 Prediction Confidence Functions. We introduce three simple yet robust functions that, given a scheme that predicts functional labels based on neighborhood voting, return a confidence value for predicted node labels. We note that these measures are of independent interest, even though the focus of the current paper is on incorporating them into cascading schemes for predicting function over multiple rounds using variants of cDSD-based majority voting.

For each confidence function, we denote $\hat{l}_i$ as the predicted label of node v_i, and L_i as the set of labels that received votes during a cDSD-MV or cDSD-WMV round. For each node v_i, each confidence function outputs a score c_i between 0 and 1 inclusive, where higher scores indicate higher confidence in the predicted label. Our three confidence functions are described as follows:

- Count-Conf **(CC)**: analogous to majority voting, where c_i is the ratio of votes for the winning label $\hat{l}_i$ and the sum of votes over all labels in L_i:

$$c_i = \frac{\#\text{ votes for } \hat{l}_i}{\sum_{l \in L_i} \#\text{ votes for } l}$$

- Weighted-Count-Conf **(WCC)**: analogous to weighted majority voting, where c_i is the ratio of the sum of weighted votes for the winning label $\hat{l}_i$ and the weighted sum of votes over all labels in L_i:

$$c_i = \frac{\text{sum of weighted votes for } \hat{l}_i}{\sum_{l \in L_i} \text{sum of weighted votes for } l}$$

- Entropy-Conf **(EC)**: in this function, we attempt to measure the 'purity' of L_i compared to a uniform distribution of votes. For all $l \in L_i$, we denote p_l as the proportion of votes for l, and u as the reciprocal of total unique labels:

$$p_l = \frac{\#\text{ of votes for } l}{\sum_{j \in L_i} \#\text{ of votes for } j} \qquad u = \frac{1}{|L_i|}$$

Then the confidence score for node v_i is given by:

$$c_i = 1 - \frac{\sum_{l \in L_i} p_l \log_2 p_l}{\sum_{l \in L_i} u \log_2 u}$$

Here, c_i will be near 0 when each label in L_i received a similar number of votes, and near 1 when a single label received most votes.

In addition to these three primary confidence functions, we also define a fourth function used primarily as a control in the cascading procedure:

- Random-conf **(RC)**: a random integer between 0 and 100 scaled between 0 and 1:

$$c_i = \frac{x \sim [0, 100]}{100}$$

2.3 Evaluation

2.3.1 Multi-fold cross-validation tasks. We consider m-fold cross validation tasks where $m = 2, 4, 6$ according to the following scheme: in each of the m-fold cross validation tasks, we first randomly split the annotated proteins into m sets. We use the annotations of nodes in each of the m sets once as a training set to predict annotations on the remaining $m - 1$ sets. We then average the performance over the m-folds of cross validation. We conduct 10 runs of each m-fold cross validation task and report means and standard deviations of our performance measures over these 10 runs.

Note that this is the opposite training-testing proportional split from the most commonly used cross-validation setups. At each fold as m increases, the number of nodes in our training set will decline (whereas in ordinary cross validation the number of training nodes increases). For example, our 2-fold CV tasks use 50% of our annotated nodes as a training set, our 4-fold CV tasks use 25% of annotated nodes as a training set, and our 6-fold CV tasks use 17% of annotated nodes as a training set. We did this deliberately to target the interesting case of having smaller and smaller labeled training sets of nodes with known labels. As an alternative to down-sampling known labels, this allows us to understand the effectiveness of the cascading procedure when the full network topology is known, but the amount of known annotations is limited. This is a particularly important scenario when trying to design function prediction mechanisms that will still be effective in less well-studied and therefore less well-annotated organisms.

For both GO and MIPS annotations, we consider the following two performance measurements:

a. Prediction **accuracy**, which is calculated as the percentage of proteins that are assigned a correct annotation [18]. An annotation is considered correct if it appears among the true labels assigned to the protein.
b. **F1 score**, which is the performance suggested in [8] and calculated as:

$$\text{F1} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Here, precision and recall are calculated by looking at the top α predicted annotations of each protein, and we average F1 scores over the individual functions to obtain the overall F1 score for a round of predictions. As in [3] and [2], we set $\alpha = 3$.

3 RESULTS

3.1 Results on *S. cerevisiae*

In order to define a cascading framework, four parameters need to be set. These are: the choice of voting method, the choice of confidence function, the number of cascade rounds, and the high-confidence threshold. Table 1 reports average percent accuracy and F1 Score on the *S. cerevisiae* network across all four majority vote variants described in Section 2.2.3 using the EC confidence function, 12 cascade rounds, and a 35% high-confidence threshold (where we explain further how we tuned these parameters below). Additionally, tables that report the same statistics but using the CC and WCC confidence functions appear in the supplement. We compare this to accuracy and F1 Score using each prediction method *without* cascading, and we report over 2-folds, 4-folds, and 6-folds of cross validation for MIPS1, MIPS2, MIPS3, and GO-MFBP-4 labels.

We find that for the variants of majority vote that appeared in [2, 3] (cDSD-MV and cDSD-WMV), cascading leads to substantial accuracy improvements in all label categories and over all levels of cross validation. Additionally, we notice that accuracy improvements increase as the number of initially labeled nodes decreases (CV fold size gets larger). For example, using MIPS1 labels and the cDSD-WMV function, cascading improved accuracy by 1.15% in 2-fold CV, by 4.93% in 4-fold CV, and by 10.04% in 6-fold CV. For the two new variants of majority vote that dynamically expand the scope of each node's local neighborhood under the cDSD metric to find labeled neighbors (cDSD-MV-Known and cDSD-WMV-Known), we find that cascading improves accuracy, but by more modest levels. In some cases, like using the cDSD-WMV-Known method on MIPS1 labels with 2-fold CV, cascading leads to essentially the same level of performance as running a single round of predictions. Overall we notice that the cDSD-WMV-Known method using cascading leads to the highest average prediction accuracy across all label types and levels of CV.

Table 1: Summary of results for *S. cerevisiae* using cascading across label type, prediction method, and CV fold size. We report mean and standard deviation across 10 random splits of testing and training data. A fixed cascading setting (12 rounds, 35% high confidence threshold, EC confidence function) is used.

			2-Fold		4-Fold		6-Fold	
			acc.	F1	acc.	F1	acc.	F1
MIPS1	cDSD-MV		66.71 ± 0.57	41.43 ± 0.29	57.96 ± 0.32	40.43 ± 0.23	49.38 ± 0.31	37.86 ± 0.26
	cDSD-MV	*casc.*	67.54 ± 0.25	41.99 ± 0.20	62.30 ± 0.36	42.08 ± 0.24	59.05 ± 1.44	40.67 ± 0.21
	cDSD-MV-Known		67.65 ± 0.52	41.69 ± 0.25	64.21 ± 3.24	41.89 ± 0.15	62.34 ± 0.23	41.01 ± 0.06
	cDSD-MV-Known	*casc.*	67.96 ± 0.40	42.03 ± 0.22	65.10 ± 0.25	42.42 ± 0.14	62.80 ± 0.26	41.54 ± 0.17
	cDSD-WMV		66.71 ± 0.42	42.07 ± 0.24	57.83 ± 0.52	41.07 ± 0.22	49.06 ± 0.24	38.26 ± 0.22
	cDSD-WMV	*casc.*	67.86 ± 0.58	**42.59** ± 0.26	62.76 ± 0.63	42.41 ± 0.14	59.10 ± 0.56	40.98 ± 0.11
	cDSD-WMV-Known		68.31 ± 0.36	42.40 ± 0.31	64.91 ± 0.30	42.74 ± 0.14	63.06 ± 0.28	**41.99** ± 0.15
	cDSD-WMV-Known	*casc.*	**68.37** ± 0.46	42.34 ± 0.12	**65.43** ± 0.37	**42.77** ± 0.17	**63.63** ± 0.53	41.94 ± 0.15
MIPS2	cDSD-MV		53.50 ± 0.71	32.55 ± 0.22	44.32 ± 0.29	30.30 ± 0.09	37.29 ± 0.38	27.85 ± 0.23
	cDSD-MV	*casc.*	54.74 ± 0.55	32.83 ± 0.29	49.08 ± 0.32	31.04 ± 0.13	44.90 ± 0.34	29.01 ± 0.13
	cDSD-MV-Known		54.70 ± 0.39	32.87 ± 0.23	50.39 ± 0.18	31.70 ± 0.09	48.17 ± 0.19	30.36 ± 0.14
	cDSD-MV-Known	*casc.*	55.26 ± 0.38	33.09 ± 0.20	51.31 ± 0.24	32.08 ± 0.15	49.15 ± 0.26	30.68 ± 0.13
	cDSD-WMV		55.02 ± 0.58	33.57 ± 0.24	45.94 ± 0.47	31.38 ± 0.18	38.03 ± 0.33	28.70 ± 0.15
	cDSD-WMV	*casc.*	55.72 ± 0.59	33.96 ± 0.21	49.74 ± 0.28	32.02 ± 0.16	45.46 ± 0.38	29.85 ± 0.15
	cDSD-WMV-Known		55.85 ± 0.59	33.87 ± 0.19	51.86 ± 0.38	32.93 ± 0.17	49.42 ± 0.21	31.51 ± 0.11
	cDSD-WMV-Known	*casc.*	**56.27** ± 0.53	**34.11** ± 0.15	**52.09** ± 0.33	**33.04** ± 0.11	**49.86** ± 0.22	**31.69** ± 0.11
MIPS3	cDSD-MV		47.95 ± 0.49	27.06 ± 0.22	39.54 ± 0.28	26.76 ± 0.30	32.59 ± 0.28	25.31 ± 0.27
	cDSD-MV	*casc.*	49.18 ± 0.50	27.47 ± 0.18	44.04 ± 0.16	27.14 ± 0.18	39.95 ± 0.24	25.70 ± 0.17
	cDSD-MV-Known		49.15 ± 0.43	27.46 ± 0.17	44.42 ± 0.22	26.84 ± 0.18	41.48 ± 0.26	25.50 ± 0.11
	cDSD-MV-Known	*casc.*	49.71 ± 0.42	27.57 ± 0.19	45.66 ± 0.20	27.14 ± 0.09	42.95 ± 0.36	25.69 ± 0.13
	cDSD-WMV		50.03 ± 0.34	28.41 ± 0.24	41.31 ± 0.35	27.79 ± 0.27	34.07 ± 0.38	26.00 ± 0.27
	cDSD-WMV	*casc.*	50.81 ± 0.56	28.56 ± 0.15	44.97 ± 0.49	28.11 ± 0.17	40.72 ± 0.38	26.43 ± 0.16
	cDSD-WMV-Known		50.70 ± 0.46	28.33 ± 0.11	46.64 ± 0.27	28.13 ± 0.11	43.66 ± 0.33	26.83 ± 0.11
	cDSD-WMV-Known	*casc.*	**51.36** ± 0.39	**28.69** ± 0.21	**46.99** ± 0.18	**28.26** ± 0.17	**44.13** ± 0.38	**26.85** ± 0.17
GO-MFBP-4	cDSD-MV		47.41 ± 0.59	13.24 ± 0.18	37.98 ± 0.29	10.77 ± 0.11	31.19 ± 0.44	9.03 ± 0.07
	cDSD-MV	*casc.*	49.05 ± 0.53	13.51 ± 0.09	42.36 ± 0.27	11.40 ± 0.09	37.96 ± 0.28	9.88 ± 0.06
	cDSD-MV-Known		49.07 ± 0.57	13.68 ± 0.10	44.09 ± 0.34	12.48 ± 0.11	41.31 ± 0.18	11.76 ± 0.06
	cDSD-MV-Known	*casc.*	49.87 ± 0.57	13.77 ± 0.10	45.50 ± 0.33	12.59 ± 0.05	43.01 ± 0.17	11.75 ± 0.05
	cDSD-WMV		49.41 ± 0.41	13.67 ± 0.10	39.48 ± 0.43	11.23 ± 0.10	31.98 ± 0.36	9.35 ± 0.10
	cDSD-WMV	*casc.*	50.46 ± 0.55	13.85 ± 0.13	43.41 ± 0.54	11.69 ± 0.08	38.52 ± 0.23	10.15 ± 0.08
	cDSD-WMV-Known		50.49 ± 0.45	**14.06** ± 0.16	45.99 ± 0.36	12.93 ± 0.07	43.42 ± 0.21	**12.16** ± 0.08
	cDSD-WMV-Known	*casc.*	**51.06** ± 0.69	14.03 ± 0.15	**46.70** ± 0.36	**13.01** ± 0.11	**44.11** ± 0.33	12.14 ± 0.07

We now provide some information on how the cascading parameters used to generate the cross validation results in Table 1 were set. First we found that the EC confidence function, compared to the other confidence functions, was the most likely to generate confidence scores that correlated with correct predictions in yeast. This can be observed in Table 2, where we performed 2-fold cross validation using the cDSD-MV function *without* cascading, and used each of our three main confidence functions (and also the RC function as a control) to generate confidence scores on each prediction. Given a high-confidence threshold of 10% or 25%, we separated nodes into high-confidence and low-confidence sets, and we measured the average accuracy of predictions among the nodes in both sets independently. Here, we notice that across all label types, using the EC confidence function leads to the highest levels of accuracy among high-confidence predictions. The other two main confidence functions, CC and WCC, also show a significant margin between the accuracy of high and low confidence predictions, which is in comparison to the RC function (used as a control), which indicates no discernible difference in the prediction performance between high and low confidence nodes. These trends continue over high-confidence thresholds of 40% and 50%, and those results are found in the supplement.

We also measured how prediction accuracy was affected by increasing the high-confidence threshold or the number of cascade rounds within the cascading framework. We first evaluate the former, where we use each of our confidence functions in a cascading procedure in 2-fold, 4-fold, and 6-fold CV settings with the cDSD-MV prediction method. We use a constant 10 cascade rounds and set the high-confidence threshold to 10%, 20%, 30%, 40%, and 50%. We present plots of these results in Figure 3 for MIPS1 and GO-MFBP-4 labels, and we notice that using each of the three main

Table 2: Avg. Distribution of accuracy among high and low confidence predictions after 1 round of voting across confidence functions, using high-confidence thresholds of 10% and 25%; Results shown for *S. cerevisiae* using 2-Fold CV (50% of training data) and cDSD-MV. We report mean and standard deviation across 10 random splits of testing and training data.

		MIPS1		MIPS2		MIPS3		GO-MFBP-4	
		high-conf	*low-conf*	*high-conf*	*low-conf*	*high-conf*	*low-conf*	*high-conf*	*low-conf*
[**CC**]	10%	83.0 ± 1.1	62.8 ± 0.5	82.0 ± 0.4	46.7 ± 0.5	79.7 ± 0.9	40.9 ± 0.7	61.9 ± 1.6	44.5 ± 0.3
	25%	78.3 ± 0.8	57.5 ± 0.8	70.5 ± 0.6	39.4 ± 0.7	68.0 ± 1.0	32.5 ± 0.6	56.0 ± 1.0	41.9 ± 0.6
[**WCC**]	10%	84.5 ± 1.3	62.5 ± 0.4	84.1 ± 1.1	46.5 ± 0.5	79.7 ± 0.7	40.3 ± 0.7	63.4 ± 1.6	44.0 ± 0.8
	25%	79.4 ± 0.5	57.4 ± 0.7	72.9 ± 0.9	39.2 ± 0.5	69.6 ± 1.1	31.9 ± 0.9	56.7 ± 0.9	41.7 ± 0.7
[**EC**]	10%	87.1 ± 1.0	61.9 ± 0.5	85.9 ± 0.7	46.4 ± 0.5	83.7 ± 1.1	39.6 ± 0.6	76.7 ± 1.2	41.1 ± 0.5
	25%	79.9 ± 0.6	56.4 ± 0.6	77.1 ± 1.0	35.7 ± 0.6	75.7 ± 0.7	27.4 ± 0.7	69.5 ± 0.7	32.0 ± 0.6
[**RC**]	10%	66.5 ± 2.1	67.1 ± 0.4	53.8 ± 2.0	53.8 ± 0.5	47.7 ± 1.9	48.1 ± 0.5	48.3 ± 2.3	48.0 ± 0.6
	25%	66.8 ± 0.6	67.2 ± 0.9	53.8 ± 1.0	54.0 ± 0.7	47.9 ± 1.0	48.0 ± 0.6	47.8 ± 0.9	48.7 ± 0.9

confidence functions (CC, WCC, and EC), prediction accuracy generally increased as the threshold grew from 10% to 20%, but ultimately plateaued when the threshold went beyond 30%. This trend is present over both label types and across levels of CV.

We then measure the effect of increasing the number of cascade rounds in a similar manner: we set a constant high-confidence threshold of 30%, and using the cDSD-MV prediction method, we evaluate prediction accuracy for MIPS1 and GO-MFBP-4 labels across CV fold levels using 2, 4, 8, 12, and 15 cascade rounds. These results are found in Figure 4, and here we notice that across all levels of CV, accuracy grows sharply within the first 8 rounds of cascading, but beyond 12 rounds performance levels off. This indicates that the initial cascade rounds contribute most to accuracy improvements, but additional rounds tend to limit (but not necessarily hurt) performance. Given the results of these three analyses, we chose the cascade setting of our main cross-validation in Table 1 to use the EC confidence function with 12 cascade rounds and a 35% high-confidence threshold.

3.2 Results on *D. Melanogaster*

Given our interest in evaluating the effectiveness of cascading on less-studied organisms, we perform a similar set of cross validation tasks on the *D. melanogaster* network. Note that unlike yeast, we could not obtain reliable MIPS labels, so results are only presented using GO. First, the same cascade setting from Table 1 is used (EC confidence function, 12 rounds of cascading, 35% high-confidence threshold), and these results can be found in the supplementary tables. Here, we observe similar results as on the yeast network: cascading with the cDSD-MV and cDSD-WMV functions leads to substantial prediction accuracy improvements, while improvements seen using the cDSD-MV-Known and cDSD-WMV-Known functions are marginal (in a few cases, cascading leads to slight degradations in accuracy, but always within the standard deviation of the non-cascading result). We also cross validate on the fly network with a second cascade setting that uses the CC confidence function (rather than the EC function) with 12 rounds of cascading and a 35% high-confidence threshold. A summary of these results is found in Table 3, and we observe slightly stronger overall performance than the first setting. This means we would recommend using the CC confidence function for this sparser network.

3.2.1 Generating predictions for unlabeled nodes. We were curious to see what new functional labels we predict among nodes with no label at level 4 in the GO biological process or Molecular Function hierarchies in fly. A predicted label was generated for each unlabeled node (5,039 nodes) using all annotated nodes (3,866) as a training set. (*Note that since we consider nodes unlabeled if they do not have a GO annotation at four levels below the roots of the MF or BP categories of the GO hierarchy, some of our "unlabeled" nodes are actually nodes assigned more general GO labels*).

The CC confidence function was used in a cascade with 12 rounds and a 35% high-confidence threshold. Table 4 gives the top 20 predictions based on final label confidence score.

When we look at FlyBase entries for these genes, we immediately find that the top 5 most confident predictions (with .75 confidence) are almost certainly correct: these genes code for RNA, not proteins, and based on sequence similarity can be identified as small Cajal body-specific RNAs. These belong to the class of Small nucleolar RNAs (snoRNAs), which are small RNA molecules that primarily guide chemical modifications of other RNAs, mainly ribosomal RNAs, transfer RNAs, and small nuclear RNAs. So the label of "RNA Binding" seems correct. On the other hand, we could not find any evidence, either positive or negative, around the predictions of CG18094, CG42700 and CG12782 for the label "Response to Decreased Oxygen Levels." For CG7697, the label "Nucleic Acid Metabolic Process" is probably a correct prediction; FlyBase reports that the protein is part of a complex involved in RNA polyadenylation. However CG15102, which is predicted also to be involved in "Nucleic Acid Metabolic Process" with identical confidence as CG7697, is probably an incorrect prediction; there is some evidence on FlyBase that this protein is involved instead with response to decreased oxygen levels.

Additionally, we generate a similar set of predicted labels for the 798 unlabeled nodes in yeast using the 4345 labeled nodes to train. The same cascade setting used in fly (CC confidence function, 12

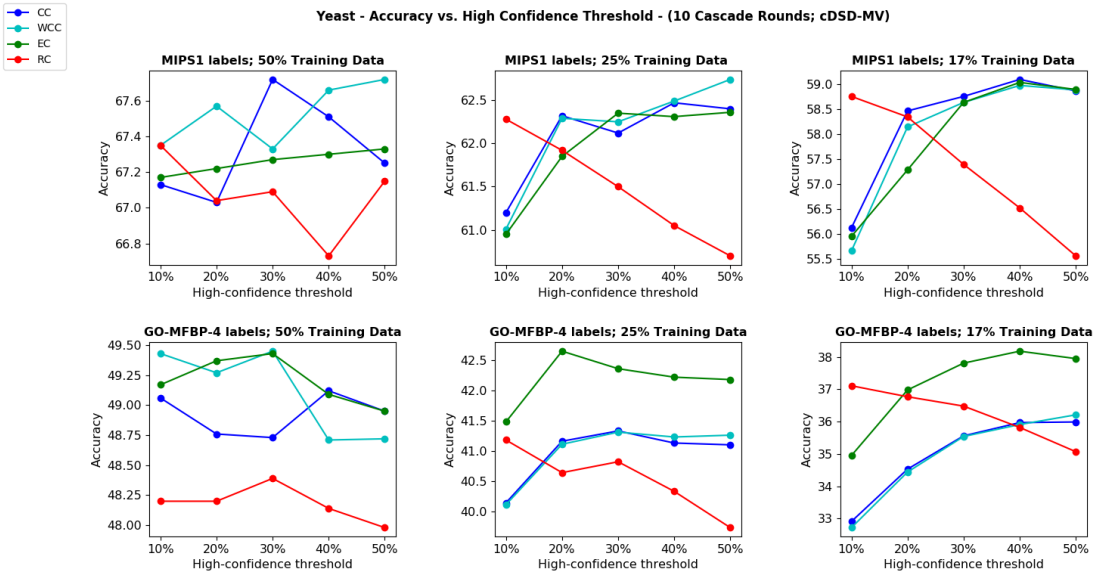

Figure 3: Prediction accuracy on *S. cerevisiae* with varying high confidence thresholds using the cDSD-MV method and 10 rounds of cascading. We see performance generally plateau (and in some instances slightly decline) at each CV fold level as the high-confidence threshold grows past 30 or 40 percent.

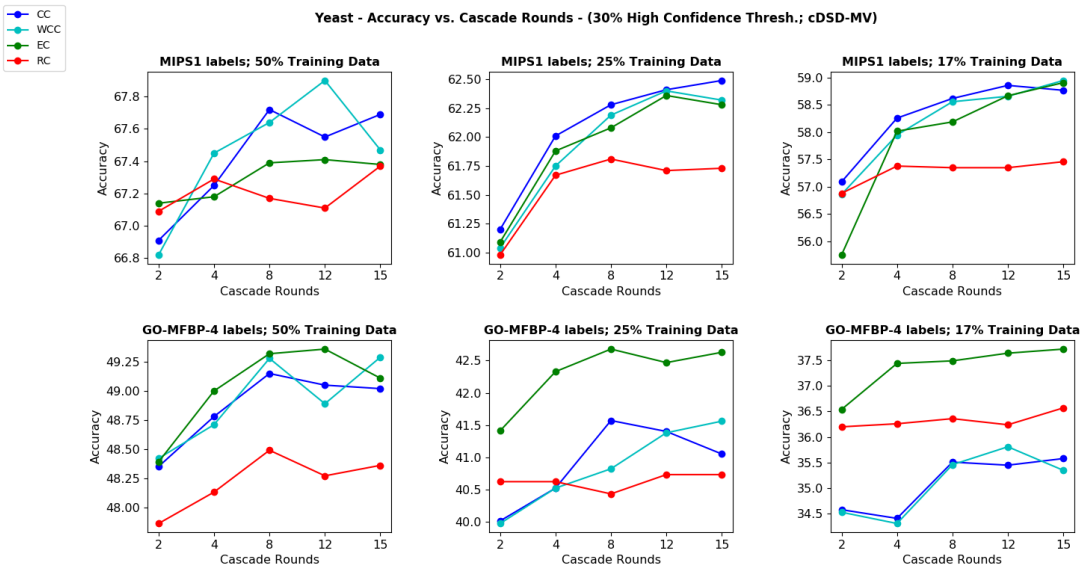

Figure 4: Prediction accuracy on *S. cerevisiae* with varying cascade rounds using the cDSD-MV method and a 35% high confidence threshold. As the number of cascade rounds increases past 12, prediction accuracy tends to level off.

cascade rounds, 35% high-confidence threshold) is used. Table 5 reports the top 20 predictions based on final label confidence score.

4 DISCUSSION

By incorporating confidence scores into existing function prediction methods, we have shown that the iterative, semi-supervised cascading framework can improve the prediction performance of existing function prediction methods in PPI networks. We observe that this performance improvement is especially significant using the original cDSD-based majority vote variants from [2, 3].

The function prediction methods used while developing and testing the cascading framework all stem from the cDSD-based

majority vote algorithms used in [2]. The results presented within this study are thus able to naturally incorporate the prediction confidence scores into subsequent rounds of predictions by allowing nodes assigned pseudo-labels to vote with weights proportional to their prediction confidence scores from previous rounds. From our analysis of tuning the number of rounds used in the cascade framework, we observe that prediction accuracy tends to plateau, rather than decrease, as the number of cascade rounds continues increasing.

This leads to several interesting questions related to the self-limiting behavior of the cascading procedure. Namely, why does performance level-off, and not sharply decrease, as we perform more cascading rounds? This could in part stem from the use of label confidence scores in voting weight when using our cDSD-based majority vote algorithms: if these confidence scores are numerically small, then even the predictions we deem as "highly-confident" in later rounds might not contribute much weight to a majority vote.

As an alternative to multiple cascading rounds, we could also imagine a scheme that runs one cascading round at a time, promoting high-confidence nodes to a confidence of 1 and demoting low-confidence nodes to a confidence of 0. Thus the high-confidence nodes assigned pseudo-labels each vote with a full weight of 1 in later rounds. It is an open question whether this would improve or hurt predictive performance.

The results in this paper are run separately for each level of MIPS, or only for GO terms at level 4 from the Biological Process or Molecular Function root nodes. However, terms at different levels of these hierarchies have ancestor-descendant relationships that could help inform function predictions. The correct way to define confidence functions that incorporate this type of hierarchical function information inheritance is an important topic for future research.

Finally, we note that results presented here only incorporate the function prediction methods of [3] and [2]. The work of [5] combined diffusion-based methods and dimensionality reduction to

Table 3: Summary of results for *D. melanogaster* using cascading across label type, prediction method, and CV fold size. We report mean and standard deviation across 10 random splits of testing and training data. A fixed cascading setting (CC confidence function, 12 rounds, 35% high confidence threshold) is used.

			2-Fold		4-Fold		6-Fold	
			acc.	F1	acc.	F1	acc.	F1
GO-MFBP-4	cDSD-MV		31.34 ± 0.57	6.23 ± 0.07	22.71 ± 0.32	5.39 ± 0.04	17.78 ± 0.16	4.52 ± 0.06
	cDSD-MV	*casc.*	33.37 ± 0.46	6.51 ± 0.08	26.03 ± 0.32	5.63 ± 0.04	21.42 ± 0.30	4.76 ± 0.04
	cDSD-MV-Known		34.62 ± 0.40	6.76 ± 0.06	32.08 ± 0.28	6.98 ± 0.07	30.96 ± 0.20	6.85 ± 0.03
	cDSD-MV-Known	*casc.*	35.39 ± 0.59	6.90 ± 0.10	32.74 ± 0.32	6.88 ± 0.05	31.45 ± 0.28	6.56 ± 0.05
	cDSD-WMV		32.20 ± 0.46	6.30 ± 0.15	23.60 ± 0.21	5.42 ± 0.05	18.30 ± 0.27	4.60 ± 0.04
	cDSD-WMV	*casc.*	33.96 ± 0.58	6.52 ± 0.13	26.54 ± 0.41	5.67 ± 0.09	21.77 ± 0.37	4.81 ± 0.05
	cDSD-WMV-Known		35.92 ± 0.50	**7.02** ± 0.05	33.23 ± 0.24	**7.19** ± 0.05	**31.72** ± 0.22	**7.08** ± 0.03
	cDSD-WMV-Known	*casc.*	**35.96** ± 0.31	6.98 ± 0.12	**33.27** ± 0.34	6.91 ± 0.06	31.62 ± 0.23	6.59 ± 0.05

Table 4: The top 20 predictions by label confidence score for unannotated nodes in the *D. melanogaster* network. Labels were predicted using cDSD-WMV-Known, the CC confidence function, 12 cascade rounds, and a 35% high-confidence threshold.

Node	Pred. Label	Pred. Label Name	Label Conf.
CR43569	GO:0003723	RNA Binding	0.750
CR32863	GO:0003723	RNA Binding	0.750
CR33716	GO:0003723	RNA Binding	0.750
CR43572	GO:0003723	RNA Binding	0.750
CR43602	GO:0003723	RNA Binding	0.750
CG18094	GO:0036293	Response to Decreased Oxygen Levels	0.375
CG42700	GO:0036293	Response to Decreased Oxygen Levels	0.333
CG12782	GO:0036293	Response to Decreased Oxygen Levels	0.333
CG8395	GO:0060255	Regulation of Macromolecule Metabolic Process	0.278
CG9606	GO:0060255	Regulation of Macromolecule Metabolic Process	0.278
CG7697	GO:0090304	Nucleic Acid Metabolic Process	0.276
CG15102	GO:0090304	Nucleic Acid Metabolic Process	0.276
CG10110	GO:0090304	Nucleic Acid Metabolic Process	0.276
CG5645	GO:0051276	Chromosome Organization	0.269
CG2202	GO:0051276	Chromosome Organization	0.269
CG34110	GO:0051276	Chromosome Organization	0.269
CG43427	GO:0051276	Chromosome Organization	0.269
CG6689	GO:0051276	Chromosome Organization	0.269
CG5618	GO:0036293	Response to Decreased Oxygen Levels	0.261
CG4585	GO:0036293	Response to Decreased Oxygen Levels	0.261

Table 5: The top 20 predictions by label confidence score for unlabeled nodes in the *S. cerevisiae* network. Labels were predicted using cDSD-WMV-Known, the CC confidence function, 12 cascade rounds, and a 35% high-confidence threshold.

Node	Pred. Label	Pred. Label Name	Label Conf.
YLR003C	GO:0090304	Nucleic Acid Metabolic Process	0.667
YOR065W	GO:0045333	Cellular Respiration	0.500
YLR336C	GO:0090304	Nucleic Acid Metabolic Process	0.417
YDR119W-A	GO:0045333	Cellular Respiration	0.350
YLR367W	GO:0090304	Nucleic Acid Metabolic Process	0.320
YKL082C	GO:0090304	Nucleic Acid Metabolic Process	0.300
YOR356W	GO:0007005	Mitochondrion Organization	0.296
YHR203C	GO:0090304	Nucleic Acid Metabolic Process	0.290
YLR287C-A	GO:0090304	Nucleic Acid Metabolic Process	0.281
YOR252W	GO:0090304	Nucleic Acid Metabolic Process	0.276
YHR172W	GO:0007010	Cytoskeleton Organization	0.259
YER002W	GO:0090304	Nucleic Acid Metabolic Process	0.258
YNL096C	GO:0090304	Nucleic Acid Metabolic Process	0.257
YGR283C	GO:0090304	Nucleic Acid Metabolic Process	0.250
YOR182C	GO:0090304	Nucleic Acid Metabolic Process	0.242
YGR027C	GO:0090304	Nucleic Acid Metabolic Process	0.219
YNL132W	GO:0090304	Nucleic Acid Metabolic Process	0.214
YDR115W	GO:0065003	Protein-Containing Complex Assembly	0.208
YDR012W	GO:0003723	RNA Binding	0.206
YOR369C	GO:0090304	Nucleic Acid Metabolic Process	0.205

improve prediction performance over standalone, DSD-based majority vote methods into a model called Diffusion Component Analysis (DCA) [5]. It is natural to ask if there is a setting where DCA can also be integrated into a cascading framework and whether this would improve that underlying method as well. A first step would be to determine the best way to compute confidences on DCA predictions.

5 ACKNOWLEDGEMENTS

We thank the Tufts BCB group for helpful discussions. This research was partially supported by NSF grant DMS-1812503 (to LC). We thank Lord Laidlaw and the Laidlaw Scholars program at Tufts for supporting the work of co-author Jonathan Rodríguez.

REFERENCES

[1] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. 2000. Gene ontology: tool for the unification of biology. *Nature genetics* 25, 1 (2000), 25.

[2] Mengfei Cao, Christopher M Pietras, Xian Feng, Kathryn J Doroschak, Thomas Schaffner, Jisoo Park, Hao Zhang, Lenore J Cowen, and Benjamin J Hescott. 2014. New directions for diffusion-based network prediction of protein function: incorporating pathways with confidence. *Bioinformatics* 30, 12 (2014), i219–i227.

[3] M Cao, H Zhang, J Park, NM Daniels, ME Crovella, et al. 2013. Going the Distance for Protein Function Prediction: A New Distance Metric for Protein. (2013).

[4] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. 2010. *Semi-Supervised Learning* (1st ed.). The MIT Press.

[5] Hyunghoon Cho, Bonnie Berger, and Jian Peng. 2015. Diffusion component analysis: unraveling functional topology in biological networks. In *International Conference on Research in Computational Molecular Biology*. Springer, 62–64.

[6] Hon Nian Chua, Wing-Kin Sung, and Limsoon Wong. 2006. Exploiting indirect neighbours and topological weight to predict protein function from protein–protein interactions. *Bioinformatics* 22, 13 (2006), 1623–1630.

[7] Lenore Cowen, Trey Ideker, Benjamin J Raphael, and Roded Sharan. 2017. Network propagation: a universal amplifier of genetic associations. *Nature Reviews Genetics* 18, 9 (2017), 551.

[8] Steven J Darnell, David Page, and Julie C Mitchell. 2007. An automated decision-tree approach to predicting protein interaction hot spots. *Proteins: Structure, Function, and Bioinformatics* 68, 4 (2007), 813–823.

[9] Minghua Deng, Zhidong Tu, Fengzhu Sun, and Ting Chen. 2004. Mapping gene ontology to proteins based on protein–protein interaction data. *Bioinformatics* 20, 6 (2004), 895–902.

[10] Minghua Deng, Kui Zhang, Shipra Mehta, Ting Chen, and Fengzhu Sun. 2003. Prediction of protein function using protein–protein interaction data. *Journal of computational biology* 10, 6 (2003), 947–960.

[11] Haretsugu Hishigaki, Kenta Nakai, Toshihide Ono, Akira Tanigami, and Toshihisa Takagi. 2001. Assessment of prediction accuracy of protein function from protein–protein interaction data. *Yeast* 18, 6 (2001), 523–531.

[12] Justin K Huang, Daniel E Carlin, Michael Ku Yu, Wei Zhang, Jason F Kreisberg, Pablo Tamayo, and Trey Ideker. 2018. Systematic evaluation of molecular networks for discovery of disease genes. *Cell systems* 6, 4 (2018), 484–495.

[13] Stanley Letovsky and Simon Kasif. 2003. Predicting protein function from protein/protein interaction data: a probabilistic approach. *Bioinformatics* 19, suppl_1 (2003), i197–i204.

[14] Pavan Kumar Mallapragada, Rong Jin, Anil K Jain, and Yi Liu. 2009. Semiboost: Boosting for semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence* 31, 11 (2009), 2000–2014.

[15] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* (2001), 415–444.

[16] Elena Nabieva, Kam Jim, Amit Agarwal, Bernard Chazelle, and Mona Singh. 2005. Whole-proteome prediction of protein function via graph-theoretic analysis of interaction maps. *Bioinformatics* 21, suppl_1 (2005), i302–i310.

[17] Andreas Ruepp, Alfred Zollner, Dieter Maier, Kaj Albermann, Jean Hani, Martin Mokrejs, Igor Tetko, Ulrich Güldener, Gertrud Mannhaupt, Martin Münsterkötter, et al. 2004. The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic acids research* 32, 18 (2004), 5539–5545.

[18] Benno Schwikowski, Peter Uetz, and Stanley Fields. 2000. A network of protein–protein interactions in yeast. *Nature biotechnology* 18, 12 (2000), 1257.

[19] Roded Sharan, Igor Ulitsky, and Ron Shamir. 2007. Network-based prediction of protein function. *Molecular systems biology* 3, 1 (2007), 88.

[20] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. 2006. BioGRID: a general repository for interaction datasets. *Nucleic acids research* 34, suppl_1 (2006), D535–D539.

[21] Damian Szklarczyk, Annika L Gable, David Lyon, Alexander Junge, Stefan Wyder, Jaime Huerta-Cepas, Milan Simonovic, Nadezhda T Doncheva, John H Morris, Peer Bork, et al. 2018. STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic acids research* 47, D1 (2018), D607–D613.

[22] Alexei Vazquez, Alessandro Flammini, Amos Maritan, and Alessandro Vespignani. 2003. Global protein function prediction from protein-protein interaction networks. *Nature biotechnology* 21, 6 (2003), 697.

[23] Zhi-Hua Zhou. 2012. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC.