A Collective Learning Framework to Boost GNN Expressiveness for Node Classification

Mengyue Hang [1] **Jennifer Neville** [1] **Bruno Ribero** [1]

Abstract

Collective Inference (CI) is a procedure designed to boost weak relational classifiers, specially for node classification tasks. Graph Neural Networks (GNNs) are strong classifiers that have been used with great success. Unfortunately, most existing practical GNNs are not most-expressive (universal). Thus, it is an open question whether one can improve strong relational node classifiers, such as GNNs, with CI. In this work, we investigate this question and propose *collective learning* for GNNs —a general collective classification approach for node representation learning that increases their representation power. We show that previous attempts to incorporate CI into GNNs fail to boost their expressiveness because they do not adapt CI's Monte Carlo sampling to representation learning. We evaluate our proposed framework with a variety of state-of-the-art GNNs. Our experiments show a consistent, significant boost in node classification accuracy —regardless of the choice of underlying GNN— for inductive node classification in partially-labeled graphs, across five real-world network datasets.

1. Introduction

A large body of work in relational learning focuses on *collective classification* frameworks for strengthening poorly-expressive (i.e., local) relational node classifiers (e.g., relational logistic regression, naive Bayes, decision trees (Neville et al., 2003a)), by incorporating dependencies among node labels and propagating inferences during classification to improve performance, particularly in semi-supervised settings (Koller et al., 2007; Pfeiffer III et al., 2015; Xiang & Neville, 2008). However, a long-standing open question is *when/if collective inference is needed*, particularly as more expressive relational graph models become available, e.g., Graph Neural Networks (GNNs).

Despite the recent success of GNNs at node and graph classification tasks (Hamilton et al., 2017; Kipf & Welling, 2016; Luan et al., 2019; Xu et al., 2018), these GNNs are no more powerful than the Weisfeiler-Lehman (WL) graph isomorphism test, and thus, inherit its shortcomings. In other words, existing GNNs are not universal (most-expressive) graph representations (Chen et al., 2019; Morris et al., 2019; Murphy et al., 2019; Xu et al., 2018). This implies that these GNNs (which we refer to as WL-GNNs and also includes GCNs (Kipf & Welling, 2016)) are not expressive enough for some node classification tasks, since their representation can provably fail to distinguish non-isomorphic nodes with different labels.

While recently there has been increasing interest in developing more expressive WL-GNNs for graph classification tasks that can differentiate non-isomorphic graphs by considering higher-order GNNs (e.g. (Maron et al., 2019a; Bouritsas et al., 2020; Vignac et al., 2020; Azizian & Lelarge, 2021; Beaini et al., 2020), these methods primarily consider graph-level representations and, even when they can be adapted for node-level classification tasks, they would be computationally expensive to apply. Is there a easy-to-implement add-on procedure to existing WL-GNNs that can boost their node classification expressiveness?

To address this question, in this work, we theoretically and empirically investigate the potential for collective inference to improve the expressiveness of GNNs. We devise an add-on training and inference procedure, which we denote *collective learning*, that incorporates label dependencies among neighboring nodes via predicted label sampling—akin to how collective classification improves not-so-expressive classifiers—and show that it can improve the expressiveness of *any* WL-GNN.

Contributions:

- We propose *CL+GNN*, an add-on collective learning framework to GNNs that provably boosts their expressiveness for node classification tasks, beyond that of an *optimal* WL-GNN*. *CL+GNN* uses self-supervised

[1]Department of Computer Science, Purdue University, West Lafayette, Indiana, USA. Correspondence to: Mengyue Hang <hangm@purdue.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

*We use the term optimal WL-GNN to refer to the most ex-

learning and Monte Carlo sampled embeddings to incorporate node labels during inductive learning—and it can be implemented with *any* component GNN.

- We provide theoretical analysis of *CL+GNN*.
  - Theorem 1 shows that collective classification is provably unnecessary for GNNs that are most-expressive.
  - Since WL-GNNs are not most-expressive, Theorem 2 and Proposition 1 show that *CL+GNN* boosts the expressiveness of optimal WL-GNN and practical WL-GNNs.
  - Corollary 1 shows that previous attempts to incorporate collective inference into WL-GNNs (which in contrast to *CL+GNN* do not Monte Carlo sample embeddings) *cannot* increase expressivity beyond that of an optimal WL-GNN.
- We design and conduct extensive experiments to confirm the above theoretical claims. *CL+GNN* achieves a consistent improvement of node classification accuracy, across a variety of state-of-the-art WL-GNNs, for tasks involving unlabeled and partially-labeled test graphs. Our ablation study demonstrates the effectiveness of our approach incorporating collective learning in GNNs via self-supervised learning with Monte Carlo sampling of embeddings.

2. Problem Formulation

We consider the problem of *inductive* node classification across partially-labeled graphs, which takes as input a graph $G^{(\text{tr})} = (V^{(\text{tr})}, E^{(\text{tr})}, \boldsymbol{X}^{(\text{tr})}, \boldsymbol{Y}_L^{(\text{tr})})$ for training, where $V^{(\text{tr})}$ is a set of $n^{(\text{tr})}$ vertices, $E^{(\text{tr})} \subset V^{(\text{tr})} \times V^{(\text{tr})}$ is a set of edges with adjacency matrix $\boldsymbol{A}^{(\text{tr})}$, $\boldsymbol{X}^{(\text{tr})}$ is a $n^{(\text{tr})} \times p$ matrix containing node attributes as p-dimensional vectors, and $\boldsymbol{Y}_L^{(\text{tr})}$ is a set of observed labels (with C classes) of a connected set of nodes $V_L^{(\text{tr})} \subset V^{(\text{tr})}$, where $V_L^{(\text{tr})}$ is assumed to be a proper subset of $V^{(\text{tr})}$, noting that $V_L^{(\text{tr})} \neq \emptyset$. Let $\boldsymbol{Y}_U^{(\text{tr})}$ be the unknown labels of nodes $V_U^{(\text{tr})} = V^{(\text{tr})} \setminus V_L^{(\text{tr})}$. The goal is to learn a *joint* model of $\boldsymbol{Y}_U^{(\text{tr})} \sim P(\boldsymbol{Y}_U | G^{(\text{tr})})$ and apply this same model to predict hidden labels $\boldsymbol{Y}_U^{(\text{te})}$ in another test graph $G^{(\text{te})}$, i.e., $\hat{\boldsymbol{Y}}_U^{(\text{te})} = \arg\max_{\boldsymbol{Y}_U} P(\boldsymbol{Y}_U | G^{(\text{te})})$. The test graph $G^{(\text{te})}$ can be partially labeled or unlabeled so $V_L^{(\text{te})} \supseteq \emptyset$.

Graph Neural Networks (GNNs), which aggregate node attribute information to produce node representations, have been successfully used for this task. At the same time, relational machine learning (RML) methods, which use collective inference to boost the performance of local node classifiers via (predicted) label dependencies, have also been successfully applied to this task.

Since state-of-the-art GNNs are not most-expressive for node classification (Morris et al., 2019; Xu et al., 2018), collective classification ideas may help to improve the expressiveness of GNNs. In particular, collective inference methods often *sample* predicted labels (conditioned on observed labels) to improve the local representation around nodes and approximate the joint distribution $P(\boldsymbol{Y}_U | G^{(\text{te})})$. We also know from recent research that sampling randomized features can boost GNN expressiveness (Murphy et al., 2019). This leads to the key conjecture of this work Hypothesis 1, which we prove theoretically in Section 4 and validate empirically by extensive experimentation in Section 5.

Hypothesis 1. *Since current Graph Neural Networks (e.g. GCN, GraphSAGE, TK-GCN) cannot produce most expressive graph representations, collective learning (which takes label dependencies into account via Monte Carlo sampling) can improve the accuracy of node classification by producing a more expressive graph representation.*

Why? Because WL-GNNs can extract more information about local neighborhood dependencies via sampling (Murphy et al., 2019), and sampling predicted labels allows GNNs to pay attention to the relationship between node attributes, the graph topology, and label dependencies in local neighborhoods. With *collective learning*, GNNs will be able to incorporate more information into the estimated joint label distribution. Next, we describe our *collective learning* framework.

3. Proposed Framework: *Collective Learning*

In this section, we outline *CL+GNN*. It is a general framework to incorporate any GNN, and combines *self-supervised learning approach* and *Monte Carlo embedding sampling* in an *iterative* process to improve inductive learning on partially labeled graphs.

Specifically, given a partially labeled training graph $G^{(\text{tr})} = (V^{(\text{tr})}, E^{(\text{tr})}, \boldsymbol{X}^{(\text{tr})}, \boldsymbol{Y}_L^{(\text{tr})})$ with adjacency matrix $\boldsymbol{A}^{(\text{tr})}$ and a partially-labeled test graph $G^{(\text{te})} = (V^{(\text{te})}, E^{(\text{te})}, \boldsymbol{X}^{(\text{te})}, \boldsymbol{Y}_L^{(\text{te})})$ with adjacency matrix $\boldsymbol{A}^{(\text{te})}$. The goal of inductive node classification task is to train a joint model on $G^{(\text{tr})}$ to learn $P(\boldsymbol{Y}_U | G^{(\text{tr})})$ and apply it to $G^{(\text{te})}$ by replacing the input graph $G^{(\text{tr})}$ with $G^{(\text{te})}$. Suppose the graphs $G^{(\text{tr})}$ and $G^{(\text{te})}$, we can define $\boldsymbol{Y}_L^{(\text{tr})}$ as a binary (0-1) matrix of dimension $|V^{(\text{tr})}| \times C$, and $\boldsymbol{Y}_L^{(\text{te})}$ of dimension $|V^{(\text{te})}| \times C$, where the rows corresponding to the one-hot encoding of the (available) labels.

(Background) GNN and representation learning. Given a partially labeled graphs $G^{(\text{tr})}$, WL-GNNs generate node representation by propagating feature information through-

pressive version of a GNN–one that has the same distinguishing power as a Weisfeiler-Lehman test. Note this is not a universal graph representation.

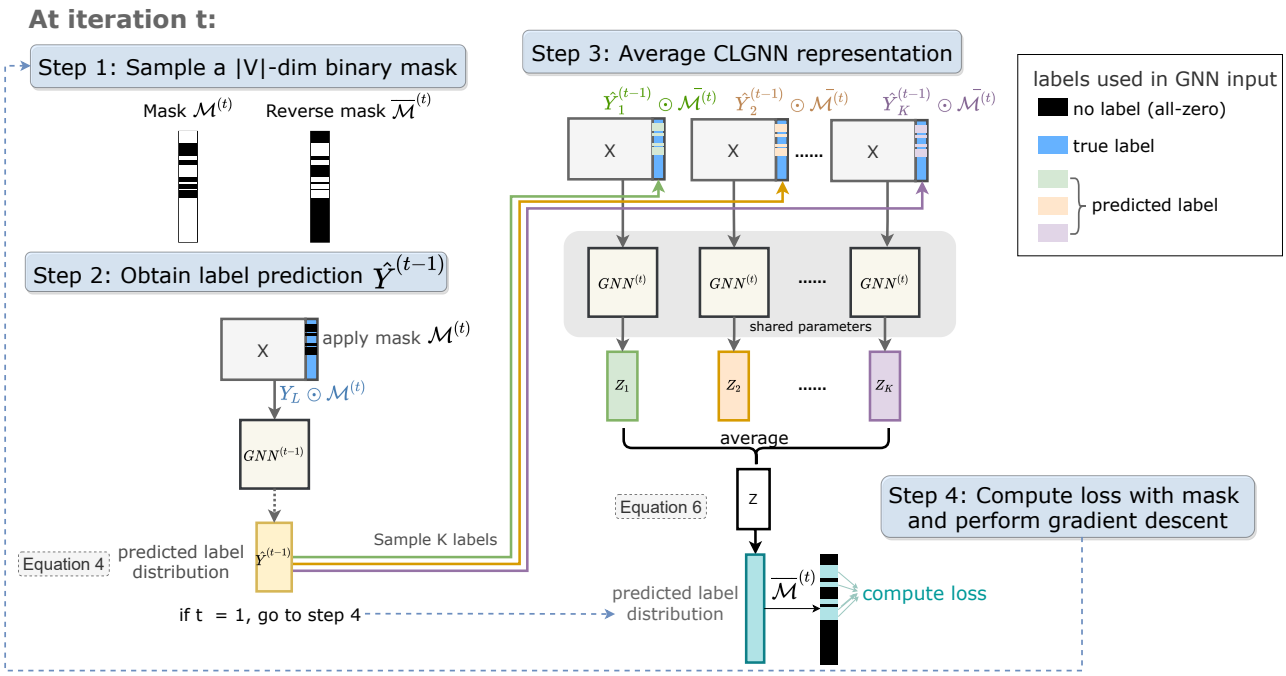

Figure 1: CLGNN model framework. Each iteration consists of four steps: (**Step 1**) Sample a random mask; (**Step 2**) Obtain predicted label distribution using the WL-GNN structure; (**Step 3**) Sample predicted labels for whatever nodes are masked, use again as input to the WL-GNN and average representations over the sampled predicted labels; (**Step 4**) Perform one optimization step by minimizing a negative log-likelihood upper bound.

out the graph. Specifically, $\forall v \in V^{(\text{tr})}$,

$$P(\boldsymbol{Y}_v|\boldsymbol{X}^{(\text{tr})}, \boldsymbol{Y}_L^{(\text{tr})}, \boldsymbol{A}^{(\text{tr})}) = \sigma(\mathbf{W}\boldsymbol{Z}_v + \mathbf{b}), \qquad (1)$$

where $\boldsymbol{Z}_v = \text{GNN}(\boldsymbol{X}^{(\text{tr})}, \boldsymbol{A}^{(\text{tr})}; \Theta)_v$ is the GNN representation of node v, $\sigma(\cdot)$ is the softmax activation, and Θ, $\mathbf{W}$ and $\mathbf{b}$ are model parameters, which are learned by minimizing the cross-entropy loss between true labels $\boldsymbol{Y}_L^{(\text{tr})}$ and the predicted labels.

The collective learning framework. Following Hypothesis 1, we propose Collective Learning GNNs (*CL+GNN*), which includes label information as input to GNNs to produce a more expressive representation. The overall framework follows four steps: (**Step 1**) Sample a random binary mask to include true labels (if available) in the input; (**Step 2**) Obtain predicted label distribution using the WL-GNN structure; (**Step 3**) Sample predicted labels for whatever nodes are masked, combine with available true labels (if any), and use again as input to the WL-GNN; finally average representations of the WL-GNN over the sampled predicted labels; (**Step 4**) Perform one optimization step by minimizing a negative log-likelihood upper bound. These steps are shown in Figure 1. Collective learning for WL-GNNs then consists of iterating over Steps 1-4 for $t = 1, \ldots, T$ iterations. Finally, once optimized, we perform inference via Monte Carlo estimates.

***CL+GNN* loss and its representation averaging.** The input to GNNs is typically the full graph $G^{(\text{tr})}$. If we included the observed labels $\boldsymbol{Y}_L^{(\text{tr})}$ directly in the input, then it would be trivial to learn a model that predicts part of the input. Instead, we either (*scenario test-unlabeled*) mask all label inputs if the test graph $G^{(\text{te})}$ is expected to be unlabeled; or (*scenario test-partial*) if $G^{(\text{te})}$ is expected to have partial labels, we apply a mask to the labels we wish to predict in training so they do not appear in the input $\boldsymbol{Y}_L^{(\text{tr})}$.

Specifically, at the t-th step of our optimization —these steps can be coarser than a gradient step — we either (*scenario test-partial*) sample a mask $\boldsymbol{M}^{(t)} \sim \text{Uniform}(\mathcal{M})$ or (*scenario test-unlabeled*) set $\boldsymbol{M}^{(t)} = \mathbf{0}$. For now, we assume we can sample $\hat{\mathbf{Y}}^{(t-1)} = (\hat{\boldsymbol{Y}}_v^{(t-1)})_{v \in V^{(\text{tr})}}$ from an estimate of the distribution $P(\boldsymbol{Y}_v^{(\text{tr})}|\boldsymbol{X}^{(\text{tr})}, \boldsymbol{Y}_L^{(\text{tr})} \odot \boldsymbol{M}^{(t)}, \boldsymbol{A}^{(\text{tr})})$ —we will come back to this assumption soon. Let $\boldsymbol{X}^{(\text{tr})}_{\boldsymbol{Y}_L^{(\text{tr})}, \hat{\boldsymbol{Y}}^{(t-1)}, \boldsymbol{M}^{(t)}}$ be the matrix concatenation between $\boldsymbol{Y}_L^{(\text{tr})} \odot \boldsymbol{M}^{(t)} + \hat{\boldsymbol{Y}}^{(t-1)} \odot \overline{\boldsymbol{M}}^{(t)}$ and $\boldsymbol{X}^{(\text{tr})}$, where again $\overline{\boldsymbol{M}} := \mathbf{1} - \boldsymbol{M}$ is the bitwise negated matrix of $\boldsymbol{M}$. Let

$$\boldsymbol{Z}_v^{(t)}(\boldsymbol{M}^{(t)}; \Theta) = \mathbb{E}_{\hat{\mathbf{Y}}^{(t-1)}}\left[\text{GNN}(\boldsymbol{X}^{(\text{tr})}_{\boldsymbol{Y}_L^{(\text{tr})}, \hat{\boldsymbol{Y}}^{(t-1)}, \boldsymbol{M}^{(t)}}, \boldsymbol{A}^{(\text{tr})}; \Theta)_v\right], \qquad (2)$$

where GNN represents an arbitrary graph neural network model and Z_v^t is the *CL+GNN*'s representation obtained for node $v \in V^{(\text{tr})}$ at step $t \geq 1$.

Our optimization is defined over the expectation of $\boldsymbol{Z}_v^{(t)}(\boldsymbol{M}^{(t)})$ w.r.t. to the sampled predicted labels $\hat{\boldsymbol{Y}}^{(t-1)}$ (Equation (2)) and over a loss averaged over all sampled masks (noting that the case where $\boldsymbol{M}^{(t)} = \mathbf{0}$ is trivial):

$$\Theta_t, \mathbf{W}_t, \mathbf{b}_t = \arg\max_{\Theta,\mathbf{W},\mathbf{b}} \mathbb{E}_{\boldsymbol{M}^{(t)}} \Bigg[\sum_{v \in V_L^{(\text{tr})}} \overline{\boldsymbol{M}}_v^{(t)} \times \log \sigma(\mathbf{W} \boldsymbol{Z}_v^{(t)}(\boldsymbol{M}^{(t)}; \Theta) + \mathbf{b})_{y_v^{(\text{tr})}} \Bigg], \tag{3}$$

where again, $\sigma(\cdot)$ is the softmax activation function, and $V_L^{(\text{tr})}$ are the labeled nodes in training graph.

Stochastic optimization of Equation (3). Equation (3) is based on a pseudolikelihood, where the joint distribution of the labels $\{\boldsymbol{Y}_v^{(\text{tr})} : v \in V_L^{(\text{tr})} \text{ s.t. } \overline{\boldsymbol{M}}_v^{(t)} = 1\}$ is decomposed as marginal distributions resulting in the sum over $V_L^{(\text{tr})}$.

(Step 1) Sample a binary mask In (*scenario test-partial*), where $G^{(\text{te})}$ is expected to have some observed labels, we randomly sample a binary mask $\boldsymbol{M} \sim \text{Uniform}(\mathcal{M})$ from a set of masks, where $\boldsymbol{M}$ is a $|V^{(\text{tr})}| \times C$ binary (0-1) matrix with the same $|V^{(\text{tr})}|$-dimensional vector in each column. By applying the mask on the observed labels $\boldsymbol{Y}_L^{(\text{tr})}$, the set of true labels is effectively partitioned into two parts, where part of the true labels $\boldsymbol{Y}_L^{(\text{tr})} \odot \boldsymbol{M}$ are used as input to *CL+GNN*, and the other part $\boldsymbol{Y}_L^{(\text{tr})} \odot \overline{\boldsymbol{M}}$ are used as optimization target. Here $\overline{\boldsymbol{M}} := \mathbf{1} - \boldsymbol{M}$ is the bitwise negated matrix of $\boldsymbol{M}$.

(Step 2) Obtaining $\hat{\mathbf{Y}}^{(t-1)}$. Note that in Equation (2), we first need to obtain the predicted label distribution $\hat{\mathbf{Y}}^{(t-1)}$ with mask $M^{(t)}$ to sample labels from. At iteration t, we use the learned *CL+GNN* model parameter Θ_{t-1} to obtain $\boldsymbol{Z}_v^{(t-1)}$ according to Equation (2) and use the *CL+GNN* model parameters $\mathbf{W}_{t-1}, \mathbf{b}_{t-1}$ to obtain the label prediction recursively, i.e. $\forall v \in V^{(\text{tr})}$,

$$\hat{\boldsymbol{Y}}_v^{(t-1)} \sim \text{Categorical}(\sigma(\mathbf{W}_{t-1} \boldsymbol{Z}_v^{(t-1)}(\boldsymbol{M}^{(t)}; \Theta_{t-1}) + \mathbf{b}_{t-1})), \tag{4}$$

where

$$\boldsymbol{Z}_v^{(t-1)}(\boldsymbol{M}^{(t)}; \Theta_{t-1}) = \text{GNN}(\boldsymbol{X}^{(\text{tr})}_{\boldsymbol{Y}_L^{(\text{tr})}, \mathbf{0}, \boldsymbol{M}^{(t)}}, \boldsymbol{A}^{(\text{tr})}; \Theta_{t-1})_v \tag{5}$$

Note that $\boldsymbol{Z}_v^{(t-1)}(\boldsymbol{M}^{(t)}; \Theta)$ does not use any predicted labels in the GNN input, i.e. it uses the true labels for masked nodes or all-zero labels for unmasked nodes.

In order to optimize Equation (3), we compute gradient estimates w.r.t. Θ and $\mathbf{b}$ using the following sampling procedure.

(Step 3) We first need to compute an unbiased estimate of $\{\boldsymbol{Z}_v^{(t-1)}\}_{v \in V_L^{(\text{tr})}}$ in Equation (2) using K i.i.d. samples $\hat{\boldsymbol{Y}}^{(t-1)}$ from the model obtained at time step $t-1$ (as describe above), i.e.

$$\tilde{\boldsymbol{Z}}_v^{(t)}(\boldsymbol{M}^{(t)}; \Theta_t) = \frac{1}{K} \sum_{k=1}^{K} \text{GNN}(\boldsymbol{X}^{(\text{tr})}_{\boldsymbol{Y}_L^{(\text{tr})}, \hat{\boldsymbol{Y}}_k^{(t-1)}, \boldsymbol{M}^{(t)}}, \boldsymbol{A}^{(\text{tr})}; \Theta_t)_v, \tag{6}$$

where again $\boldsymbol{X}^{(\text{tr})}_{\boldsymbol{Y}_L^{(\text{tr})}, \hat{\boldsymbol{Y}}^{(t-1)}, \boldsymbol{M}^{(t)}}$ is the matrix concatenation between $\boldsymbol{X}^{(\text{tr})}$ and $\boldsymbol{Y}_L^{(\text{tr})} \odot \boldsymbol{M}^{(t)} + \hat{\boldsymbol{Y}}^{(t-1)} \odot \overline{\boldsymbol{M}}^{(t)}$.

Note that the time/space complexity of the *CL+GNN* is K times the time/space complexity of the corresponding GNN model as we have to compute K representations for each node at each stochastic gradient step.

(Step 4) Next, we need an unbiased estimate of the expectation over mask $\boldsymbol{M}^{(t)}$ in Equation (3). In *(scenario test-partial)* the unbiased estimates are obtained by sampling $\boldsymbol{M}^{(t)} \sim \text{Uniform}(\mathcal{M})$ at each gradient step, in the *(scenario test-unlabeled)* the value obtained is exact since $\boldsymbol{M}^{(t)} = \mathbf{0}$. The mask $\boldsymbol{M}^{(t)}$ is used, along with the estimate $\tilde{\boldsymbol{Z}}$ from Equation (6), to compute the loss function as in Equation (3) and perform a gradient descent step. Proposition 2 shows that the above procedure is a proper surrogate upperbound of the loss function.

Inference with learned model.

Once the *CL+GNN* parameters $\Theta_T, \boldsymbol{W}_T, \boldsymbol{b}_T$ are learned according to Equation (3) on the training graph $G^{(\text{tr})}$, given an any-size attributed graph $G^{(te)}$, we sample J masks $\boldsymbol{M}$ of size $|V^{(\text{te})}|$, either (*scenario test-partial*) sampling $\boldsymbol{M} \sim \text{Uniform}(\mathcal{M})$ or (*scenario test-unlabeled*) set $\boldsymbol{M} = \mathbf{0}$. For each mask, we apply the same procedure as in **(Step 2)** and **(Step 3)** to obtain predicted label distribution $\hat{\mathbf{Y}}^{(\text{tmp})}$, and then sample K labels $\{\hat{\mathbf{Y}}_1^{(\text{tmp})}, \cdots, \hat{\mathbf{Y}}_K^{(\text{tmp})}\}$ from it and pass to the learned model. The node representations for $v \in V^{(\text{te})}$ are obtained using $\boldsymbol{M}$ and $\hat{\mathbf{Y}}^{(\text{tmp})}_{1,\cdots,K}$:

$$\tilde{\boldsymbol{Z}}_v(M; \Theta_T) = \frac{1}{K} \sum_{k=1}^{K} \text{GNN}(\boldsymbol{X}^{(\text{te})}_{\boldsymbol{Y}_L^{(\text{te})}, \hat{\boldsymbol{Y}}_k^{(\text{tmp})}, \boldsymbol{M}}, \boldsymbol{A}^{(\text{te})}; \Theta_T)_v,$$

where

$$(\hat{\boldsymbol{Y}}_k^{(\text{tmp})})_v \sim \text{Categorical}(\sigma(\mathbf{W}_T \boldsymbol{Z}_v^{(\text{tmp})}(\boldsymbol{M}; \Theta_T) + \mathbf{b}_T)),$$

and

$$\boldsymbol{Z}_v^{(\text{tmp})}(\boldsymbol{M}; \Theta_T) = \text{GNN}(\boldsymbol{X}_{\boldsymbol{Y}_L^{(\text{te})}, \mathbf{0}, \boldsymbol{M}^{(\text{te})}}, \boldsymbol{A}^{(\text{te})}; \Theta_T)_v.$$

The final node representation is computed as the average over all sampled masks:

$$\tilde{\boldsymbol{Z}}_v = \frac{1}{J} \sum_{j=1}^{J} \tilde{\boldsymbol{Z}}_v(M_j; \Theta_T),$$

where J and K are hyperparameters, J is the number of masks for our Monte Carlo average and K is the number

of Monte Carlo samples of $\hat{\mathbf{Y}}^{(\text{tmp})}$. Then the label predictions are obtained using the learned *CL+GNN* parameters $\boldsymbol{W}_T, \boldsymbol{b}_T$:

$$\hat{\boldsymbol{Y}}_v^{(\text{te})} \sim \text{Categorical}(\sigma(\mathbf{W}_T \tilde{\boldsymbol{Z}}_v + \mathbf{b}_T)_v),\ \forall v \in V^{(\text{te})}. \quad (7)$$

4. *Collective Learning* Analysis

Is collective classification able to better represent target label distributions than node representation learning? The answer to this question is both *yes (for WL-GNNs) and no (for most-expressive representations)*. Theorem 1 shows that a most-expressive graph representation (Murphy et al., 2019; Maron et al., 2019b; Srinivasan & Ribeiro, 2019) would not benefit from a collective learning boost. All proofs can be found in the Appendix.

Theorem 1 (Collective classification can be unnecessary)**.** *Consider the task of predicting node labels when no labels are available in test data. Let $\Gamma^\star(v, G^{(te)})$ be a most-expressive representation of node $v \in V^{(te)}$ in graph $G^{(te)}$. Then, for any collective learning procedure predicting the class label of $v \in V^{(te)}$, there exists a classifier that takes $\Gamma^\star(v, G^{(te)})$ as input and predicts the label of v with equal or higher accuracy.*

While Theorem 1 shows that the most-expressive graph representation does not need collective classification, WL-GNNs are not most-expressive (Morris et al., 2019; Murphy et al., 2019; Xu et al., 2018). Indeed, Theorem 2 and Proposition 1 show that *CL+GNN* boosts the expressiveness of optimal WL-GNN and practical WL-GNNs, respectivelly. Then, we show that the stochastic optimization in Step 3 optimizes a loss surrogate upper bound.

4.1. Expressive power of *CL+GNN*

Morris et al. (2019) and Xu et al. (2018) show that WL-GNNs are no more powerful in distinguishing non-isomorphic graphs and nodes as the standard Weisfeiler-Lehman graph isomorphism test (1-WL or just WL test). Two nodes are assumed isomorphic by the WL test if they have the same color assignment in the stable coloring. The node-expressivity of a parameterized graph representation Γ (with parameter $\Gamma(\cdot; \boldsymbol{W})$) can then be determined by the set of graphs for which Γ can identify non-isomorphic nodes:

$$\begin{aligned}\mathcal{G}(\Gamma) = \{G : \exists \boldsymbol{W}_G^\star,\ &\text{s.t. } \forall u, v \in V_G, \Gamma(G; \boldsymbol{W}_G^\star)_v \\ &= \Gamma(G; \boldsymbol{W}_G^\star)_u \text{ iff } u, v \text{ are isomorphic}, G \in \mathbb{G}\},\end{aligned}$$

where $\mathbb{G}$ is the set of all any-size attributed graphs, V_G is the set of nodes in graph G. We call $\mathcal{G}(\Gamma)$ the *identifiable set* of graph representation Γ.

The *most expressive* graph representation $\Gamma^\star$ has an *identifiable set* of all any-size attributed graphs, i.e. $\mathcal{G}(\Gamma^\star) = \mathbb{G}$. We refer to the WL-GNN that is equally expressive as WL test as the *optimal* WL-GNN (or WLGNN*), which is at least as expressive as all other WL-GNNs.

In this section we show that *collective learning* can boost the optimal WLGNN*, i.e., the identifiable set of WLGNN* is a proper subset of collective learning over WLGNN* (denoted *CL+GNN**)

$$\mathcal{G}(\text{WLGNN}^\star) \subsetneq \mathcal{G}(\textit{CL+GNN}^\star).$$

Theorem 2 (*CL+GNN** expressive power)**.** *Let WLGNN* be an optimal WL-GNN. Then, the collective learning representation of Equation* (2)*, using WLGNN* as the GNN component, (denoted* CL+GNN**) is strictly more expressive than this WLGNN* representation model applied to the same tasks.*

Theorem 2 answers Hypothesis 1, by showing that by incorporating collective learning and sampling procedures, *CL+GNN* can boost the expressiveness of WL-GNNs, including the optimal WLGNN*.

Corollary 1. *Consider a graph representation learning method that, at iteration t, replaces $\hat{\boldsymbol{Y}}^{(t-1)}$, in Equations* (2) *and* (4) *with a deterministic function over $\boldsymbol{Z}^{(t-1)}$, e.g., a softmax function that outputs $(P(\hat{\boldsymbol{Y}}_v^{(t-1)}|\boldsymbol{Z}^{(t-1)}))_{v \in V^{(tr)}}$. Then, such method will be no more expressive than the optimal WLGNN* and, hence, less expressive than* CL+GNN*.

Corollary 1 proves that existing collective approaches are no different than current GNN methods (hence, no boosting). More specifically, it shows that existing graph representation methods that —on the surface— may even look like *CL+GNN*, but do not perform the crucial step of sampling $(\hat{\boldsymbol{Y}}_v^{(t-1)})_{v \in V^{(\text{tr})}}$, unfortunately, are no more expressive than WL-GNNs. Examples of such methods include (Fan & Huang, 2019; Moore & Neville, 2017; Qu et al., 2019; Vijayan et al., 2018).

Next, we show the practical benefits of collective learning are even greater when the WL-GNN has limited expressive power due to limited message-passing layers.

4.2. How *CL+GNN* further expands the power of few-layer WL-GNNs

A d-layer ($d > 1$) WL-GNN will only aggregate neighborhood information within d hops of any given node (i.e., over a d-hop egonet, defined as the graph representing the connections among all nodes that are at most d hops away from the center node). In practice —mostly for computational reasons— WL-GNNs have many fewer layers than the graph's diameter D, i.e., $d < D$. For instance, GCN (Kipf & Welling, 2016) and GraphSAGE (Hamilton et al., 2017) both used $d = 2$ in their experiments. Hence, they cannot differentiate two non-isomorphic nodes that are iso-

morphic within their d-hop neighborhood. We now show that *CL+GNN* can gather $2d$-hop neighborhood information with a d-layer WL-GNN.

Proposition 1. *Let G_v^d be the d-hop egonet of a node v in graph G with diameter $D > d$. Let v_1 and v_2 be two non-isomorphic nodes whose d-hop egonets are isomorphic (i.e., $G_{v_1}^d$ is isomorphic to $G_{v_2}^d$) but $2d$-hop egonets are not isomorphic. Then, a WL-GNN representation with d layers will generate identical representations for v_1 and v_2 while* CL+GNN *is capable of giving distinct node representations.*

Proposition 1 shows that collective learning has yet another benefit: *CL+GNN* further boosts the power of WL-GNNs with limited message-passing layers by gathering neighborhood information within a larger radius. Specifically, *CL+GNN* built on a WL-GNN with d layers can enlarge the effective neighborhood radius from d to $2d$ in Equation (2) , while WL-GNN would have to stack $2d$ layers to achieve the same neighborhood radius, which in practice may cause optimization challenges (i.e., $d = 2$ is a common hyperparameter value in the literature).

4.3. Optimization of *CL+GNN*

Proposition 2. *If $\forall v \in V_L^{(tr)}$, $\nabla_\Theta(\mathbf{W}\boldsymbol{Z}_v^{(t)}(\boldsymbol{M}^{(t)};\Theta))_{y_v^{(tr)}}$ is bounded (e.g., via gradient clipping), then the optimization in Equation* (3)*, with the unbiased sampling of $\{\boldsymbol{Z}_v^{(t-1)}\}_{v\in V^{(tr)}}$ and $\boldsymbol{M}^{(t)}$ described above, results in a Robbins-Monro (Robbins & Monro, 1951) stochastic optimization algorithm that optimizes a surrogate upper bound of the loss in Equation* (3)*.*

Since the optimization objective in Equation (3) is computationally impractical, as it requires computing all possible binary masks and label predictions, Proposition 2 shows that the sampling procedures used in *CL+GNN* that considers K samples of label predictions and a random mask at each gradient step is a feasible approach of estimating an unbiased upper bound of the objective.

5. Experiments

5.1. Experiment Setup

Datasets. We use datasets of Cora, Pubmed, Friendster, Facebook, and Protein. The largest dataset (Friendster (Teixeira et al., 2019)) has 43,880 nodes, which is a social network of users where the node attributes include numerical features (e.g number of photos posted) and categorical features (e.g. gender, college, etc.) encoded as binary one-hot features. The node labels represent one of the five age groups. Please refer to Appendix E for more details.

Train/Test split. Since most datasets used to test GNNs consist of a single graph, we apply Louvain community detection algorithm (Blondel et al., 2008) to split each single graph into three clusters for training, validation, and testing respectively, and remove the edges across clusters —shown in Figure 2 (left). This mimics the inductive within-graph scenario that often occurs in real world settings, where a connected subgraph is used to learn a model to generalize the remainder of the graph —e.g., Facebook would train a model on Iceland or New Zealand and then apply it to the rest of the world, see methodology in (Bakshy et al., 2014).

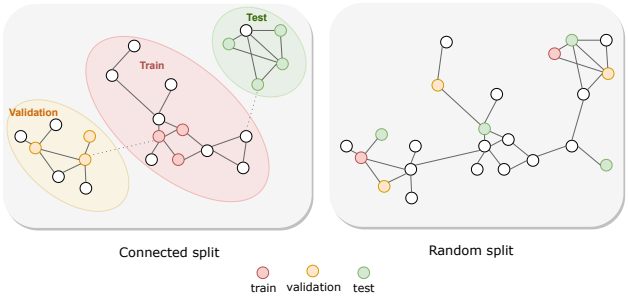

Figure 2: Different data splits between our inductive connected split (left) and conventional GNN random split (right)

Our train/test data split is different than previous GNN works, which have adopted random node split between train and test —shown in Figure 2 (right)— and can put test nodes close to the training nodes, making it much easier to leverage test node attributes during training. Our use of a hard split between train and test (connected split) is the reason why the model performance reported in our paper is not directly comparable with the reported results in previous GNN papers, even though we used the same implementations and hyper-parameter search procedures. In our experiments, we tested two different label rates in test graph: 0 (unlabeled) and 50% (reveal 50% testing labels and evaluate on the rest). We run five trials for all the experiments, and in each trial we randomly pick a connected subgraph within the training cluster and reveal their labels for training.

As our method can be applied to any GNN models, we use four representative GNNs as examples:

- GCN (Kipf & Welling, 2016) which includes two graph convolutional layers. Here we implemented an inductive variant of the original GCN model for our tasks.
- Supervised GraphSage (Hamilton et al., 2017) (denoted by GS) with Mean pooling aggregator. We use sample size of 5 for neighbor sampling.
- Truncated Krylov GCN (Luan et al., 2019) (denoted by TK), a recent GNN model that leverages multi-scale information in different ways and are scalable in depth. The TK has stronger expressive power and achieved state-of-the-art performance on node classification tasks. We implemented Snowball architecture which achieved comparable performance with the other truncated Krylov architecture according to the original paper.

		Cora$^{\text{connect}}$		Pubmed$^{\text{connect}}$		Friendster		Facebook		Protein	
# train labels:		85 (3.21%)		300 (1.52%)		641 (1.47%)		80 (1.76%)		7607 (30%)	
% labels in $G^{(\text{te})}$:		0%	50%	0%	50%	0%	50%	0%	50%	0%	50%
Random		14.28 (0.00)	14.28 (0.00)	33.33 (0.00)	33.33 (0.00)	20.00 (0.00)	20.00 (0.00)	50.00 (0.00)	50.00 (0.00)	50.00 (0.00)	50.00 (0.00)
GCN (Kipf & Welling, 2016)	-	64.74 (1.51)	66.34 (1.84)	54.56 (2.49)	58.41 (1.27)	25.97 (0.69)	24.26 (0.52)	50.58 (1.38)	51.04 (1.20)	75.86 (1.11)	77.54 (1.09)
	+ CL	**+3.72 (0.40)**	**+12.41 (1.96)**	**+1.95 (0.69)**	**+15.37 (2.01)**	**+0.70 (0.14)**	**+1.99 (0.74)**	**+2.24 (0.81)**	**+8.51 (1.09)**	**+1.22 (0.51)**	**+0.75 (0.33)**
GS (Hamilton et al., 2017)	-	65.35 (1.19)	67.71 (1.53)	55.56 (2.44)	59.12 (2.02)	26.45 (0.62)	24.75 (0.39)	51.14 (1.24)	52.06 (1.29)	73.85 (1.12)	73.01 (2.28)
	+ CL	**+2.81 (1.02)**	**+9.94(1.04)**	+1.05 (0.83)	**+14.71 (2.89)**	+0.13 (0.41)	**+1.40 (0.62)**	**+1.77 (0.55)**	**+7.80 (0.84)**	**+0.84 (0.12)**	+1.47 (0.63)
TK (Luan et al., 2019)	-	68.47 (1.31)	69.50 (0.55)	59.05 (2.13)	60.77 (1.53)	25.93 (0.91)	24.42 (1.44)	52.74 (1.62)	53.48 (1.48)	73.65 (1.69)	78.94 (1.50)
	+ CL	**+1.50 (0.61)**	**+7.92 (0.75)**	+0.23 (0.61)	**+13.62 (1.84)**	**+1.20 (0.14)**	**+2.34 (0.42)**	**+3.26 (0.98)**	**+4.60 (1.16)**	**+1.31 (0.27)**	**+1.36 (0.94)**
GRAND (Feng et al., 2020)	-	71.55 (1.07)	73.19 (0.41)	61.82 (6.40)	63.23 (7.22)	28.03 (1.02)	27.02 (0.84)	47.10 (0.27)	48.14 (0.52)	75.43 (1.12)	79.69 (0.29)
	+ CL	**+0.80 (0.31)**	**+2.30 (0.56)**	**+3.79 (1.50)**	**+5.17 (1.44)**	+0.37 (0.39)	**+4.21 (0.72)**	**+6.38 (2.29)**	**+5.72 (2.34)**	**+0.51 (0.36)**	**+0.75 (0.20)**
Best of CL		72.36 (1.20)*	78.31 (0.58)*	65.61 (6.60)*	74.39 (1.72)*	28.40 (0.85)*	31.23 (1.05)*	56.01 (1.48)	59.86 (0.83)	77.08 (1.03)	80.52 (0.37)
PL-EM (Pfeiffer III et al., 2015)	-	20.66 (0.04)	54.22 (0.94)	38.85 (0.03)	65.65 (4.33)	18.13 (0.23)	22.25 (0.87)	50.58 (0.03)	61.17 (1.14)	78.46 (1.45)	77.95 (1.56)
ICA (Lu & Getoor, 2003)	-	62.29 (2.18)	65.51 (1.30)	43.93 (6.84)	44.61 (6.24)	26.48 (1.37)	27.80 (1.56)	61.56 (1.10)*	62.04 (1.92)*	84.88 (3.35)*	84.39 (4.08)*
GMNN (Qu et al., 2019)	-	66.35 (3.12)	72.04 (2.45)	57.13 (3.01)	67.94 (4.40)	24.92 (1.20)	26.88 (1.53)	49.56 (0.88)	57.09 (0.78)	76.75 (0.74)	75.96 (0.76)

Table 1: Node classification accuracy with unlabeled and partially-labeled test data. Numbers in bold represent significant improvement in a paired t-test at the $p < 0.05$ level, and numbers with * represent the best performing method in each column. Cora$^{\text{connect}}$ and Pubmed$^{\text{connect}}$ are our processed graphs with the connected split illustrated in Figure 2 (left).

- GRAND (Feng et al., 2020), a recent GNN model using random propagation strategy to perform graph data augmentation, in order to mitigate the issues of over-smoothing and non-robustness. GRAND achieved state-of-the-art performance on several semi-supervised node classification tasks.

For each of the GNNs, we compare its baseline performance (on its own) to the performance achieved using collective learning in *CL+GNN* (using that GNN). For a fair comparison, we adopt the same hyper-parameter tuning strategy for the baseline GNNs and *CL+GNN*, e.g. hidden dimensions, learning rate, early-stopping procedures. Please refer to Appendix E for details.

In addition, we also compare to three relational classifiers, ICA (Lu & Getoor, 2003), PL-EM (Pfeiffer III et al., 2015) and GMNN (Qu et al., 2019). The first two models apply collective learning and inference with simple local classifiers —— Naive Bayes for PL-EM and Logistic regression for ICA. GMNN is the state-of-the-art collective model with GNNs, which uses two GCN models to model label dependency and node attribute dependency respectively. All the three models take true labels in their input, thus we use $\boldsymbol{Y}_L^{(\text{tr})}$ for training and $\boldsymbol{Y}_L^{(\text{te})}$ for testing.

We report the average accuracy score and standard error of five trials for the baseline models, and compute the absolute improvement of accuracy of our method over the corresponding base GNN. The best performance among all *CL+GNN* is also reported. We compute the *balanced* accuracy scores on Friendster dataset as the label is highly imbalanced. To evaluate the significance of *CL+GNN* improvements, we performed a paired t-test with five trials.

5.2. Results

The node classification accuracy of all the models is shown in Table 1. Our proposed collective learning boost is denoted as +CL (for **C**ollective **L**earning) and our model performance (absolute $\%$ of improvement over the corresponding baseline GNN) is shown in shaded area. Numbers in bold represent significant improvement over the baseline GNN based on a paired t-test ($p < 0.05$), and numbers with * is the best performing method in each column.

Comparison with baseline GNN models. Table 1 shows that our method improves the corresponding non-collective GNN models for all the four model architectures (i.e. GCN, GraphSage, TK and GRAND). Although all the models have large variances over multiple trials —because different parts of the graphs are being trained in different trials— adding CL consistently improves the baseline GNN. The results from a paired t-test comparing the performance of our method and the corresponding non-collective GNN shows that the improvement is almost always significant at $p = 0.05$ (marked as bold), with only five exceptions.

Comparing the gains on different datasets in Table 1, adding CL to GNNs achieved smaller gains on Friendster especially when no test labels were available. This is because Friendster is more sparse than the other graphs (e.g. edge density of Friendster is 1.5e-4 while Cora is 1.44e-3 (Teixeira et al., 2019)), which makes it hard for any model to propagate label information and capture label dependencies.

As expected, comparing the improvement over various GNNs with different expressive power, we observe that in general adding CL boosts the gains of simpler GNN models (i.e. GCN and GS). For example, Table 1 shows that adding CL to a GCN can boost its accuracy by +12.41% (Cora) while the boost over TK is smaller at +7.92% in the same task. This is in line with our assumption in Hypothesis 1 that collective learning can help weaker GNNs produce a more expressive representation. As GCN is less expressive than TK, there is a larger room to increase its expressiveness.

Note that the gains in Table 1 are generally much larger when we go from 0% to 50% of the labels available in

test. For example, when combining with GCN, the improvements of our method are 3.72% and 2.24% for unlabeled Cora and Facebook test sets, but with partially-labeled test data, the improvements are 12.41% and 8.51% respectively. This shows the importance of modeling label dependency especially when the some test data labels are observed.

Comparison with other relational classifiers The two baseline non-GNN relational models —i.e. PL-EM and ICA— generally perform worse than the three GNNs, with exceptions on Protein and Facebook datasets. This could be because the two dataset has only a few node attributes (3 for Facebook and 29 for Protein), while the other graphs have hundreds or thousands of node attributes, which makes it easier for the more powerful classifier (i.e. GNNs) to overfit on Facebook and Protein. Moreover, this could also be because the two non-GNN models generally need a larger portion of labeled set to train the weak local classifier, whereas GNNs utilize a neural network architecture as "local classifier", which is better at representation learning by transforming and aggregating node attribute information. However, when the model is trained with a large training set (e.g. with 30% nodes on Protein dataset), modeling the label dependency becomes crucial. At the same time, our method is still able to boost performance on the two datasets.

For GMNN (Qu et al., 2019), a collective GNN model, it achieves better performance than its non-collective base model, i.e. GCN on most of the datasets, and we can see that adding CL to GCN achieved comparable or better performance than GMNN. However, combing CL with other more powerful GNNs can easily out-perform GMNN (e.g., on Cora and Friendster, GRAND+CL significantly outperforms GMNN). When the test labels are available, GMNN is able to out-perform several GNNs by leveraging test label information, but the best of *CL+GNN* still out-performs GMNN consistently.

Ablation studies, comparison to ensembles, and hyperparameter sensitivity. We conducted three ablation studies to investigate the usage of predicted labels (detailed in Appendix F), which show that (a) adding predicted labels in model input had extra value comparing to using true labels only, (b) applying the random masking procedure is crucial for the model improvements, and (c) the gain of our framework is from using samples of the predicted labels rather than random one-hot vectors. We also compared with a baseline ensemble method, which considers an ensemble of 10 GNNs with random initialization. The results (detailed in Appendix F) show that an ensemble approach is able to slightly improve the GNN performance, but the gains are much smaller than the gains observed for *CL+GNN*.

We also investigated the impact of training labels rates and sample size K (see Appendix G), and we found that in general *CL+GNN* framework achieves a larger improvement when fewer labels are available in the training graph, and that with sample size $K > 1$ there was consistent gain.

Complexity analysis. *CL+GNN* computes K embeddings at each stochastic gradient step, therefore, per-gradient step, *CL+GNN* is K slower than its component WL-GNN. Overall, after T iterations of Steps 1-3, *CL+GNN* total runtime increases by $T \times K$ over the original runtime of its component WL-GNN. The time and space complexity of *CL+GNN* is the same as WL-GNNs, i.e. $\mathcal{O}(m)$ where $m = |E|$.

Note that existing methods trying to boost the GNN expressiveness —e.g. PPGN (Maron et al., 2019a), SMP (Vignac et al., 2020))— are much more computationally expensive in time (at least $\Theta(mn)$) and space ($\Theta(n^2)$), where n and m are number of nodes and edges in the graph.

We note that we spent nearly no time engineering *CL+GNN* for speed or for improving our results. Our interest in this paper lies entirely on the gains of a direct application of collective learning to GNNs (*CL+GNN*). We fully expect that further engineering advances can reduce the computational burden due to Monte Carlo sampling and increase accuracy gains. For instance, parallelism can significantly reduce the time to collect K samples in *CL+GNN*.

6. Related Work

On collective learning and neural networks. There has been work on applying deep learning to collective classification. For example, Moore & Neville (2017) proposed to use LSTM-based RNNs for classification tasks on graphs. They transform each node and its set of neighbors into an unordered sequence and use an RNN to predict the class label as the output of that sequence. Pham et al. (2017) designed a deep learning model for collective classification in multi-relational domains, which learns local and relational features simultaneously to encode multi-relations.

The closest work to ours is Fan & Huang (2019), which proposed a recurrent collective classification (RCC) framework, a variant of ICA (Lu & Getoor, 2003) including dynamic relational features encoding label information. Unlike our framework, this method does not sample labels $\hat{\boldsymbol{Y}}$, opting for an end-to-end training procedure. Vijayan et al. (2018) opts for a similar no-sample RCC end-to-end training method as (Fan & Huang, 2019), now combining a differentiable graph kernel with an iterative stage. Graph Markov Neural Network (GMNN) (Qu et al., 2019) is another promising approach that applies statistical relational learning to GNNs. GMNNs model the joint label distribution with a conditional random field trained with the variational EM algorithm. GMNNs are trained by alternating between an E-step and an M-step, and two WL-GCNs are trained for the two steps respectively. These studies represent different ideas for bringing the power of collective

classification to neural networks. Unfortunately, Corollary 1 shows that, without sampling $\hat{\boldsymbol{Y}}$, the above methods are still WL-GNNs, and hence, their use of collective classification fails to deliver any increase in expressiveness beyond an optimal WL-GNN (e.g., Xu et al. (2018)). In our experiments, we compared to GMNN as a representative relational GNN method, and showed that while GMNN outperformed its component GCN, the best of *CL+GNN* still consistently out-performs GMNN.

In parallel to our work, Jia & Benson (2020) considers regression tasks by modeling the joint GNN residual of a target set $(y - \hat{y})$ as a multivariate Gaussian, defining the loss function as the marginal likelihood only over labeled nodes $\hat{y}_L$. In contrast, by using the more general foundation of collective classification, our framework can seamlessly model both classification and regression tasks, and include model predictions over the entire graph $\hat{\boldsymbol{Y}}$ as *CL+GNN*'s input, thus affecting both the model prediction and the GNN training in inductive node classification tasks.

Higher-order GNNs for more expressive graph representation Recently, there has been a few works proposed to boost the representation power of WLGNN (Morris et al., 2019; Maron et al., 2019a; 2018; Vignac et al., 2020; Chen et al., 2019; Maron et al., 2019b). Most of these works consider representation for the entire graph or node sets by mimicking higher-order WL tests. However, most of them provide more theoretical implications for GNNs than practical usage due to their dependency on order-k tensors $\mathbb{R}^{n^k}$ (n : number of nodes, $k > 2$) and inability to leverage the sparsity of the graph structures. Among them PPGN (Maron et al., 2019a) is relatively scalable with $\Theta(n^3)$ time complexity and $\Theta(n^2)$ space complexity to achieve the expressive power of the 2-WL test. A more recent method SMP (Vignac et al., 2020) proposed a powerful and more scalable message-passing framework with time complexity of $\Theta(mn)$ (m : number of edges) and space complexity of $\Theta(n^2)$. Our work, on the other hand, focus on node-level representations rather than (sub)graph-level representations, and our overall time/space complexity is $\Theta(m)$. As these methods cannot be directly evaluated on node classification tasks and due to their computational inefficiency, we leave the assessment of *CL+GNN* gains if used with these more powerful GNN variants as future work.

On self-supervised learning and semi-supervised learning. Self-supervised learning is closely related to semi-supervised learning. In fact, self-supervised learning can be seen as a self-imposed semi-supervised learning task, where part of the input is masked (or transformed) and must be predicted back by the model (Doersch et al., 2015; Noroozi & Favaro, 2016; Lee et al., 2017; Misra et al., 2016). Recently, self-supervised learning has been broadly applied to achieve state-of-the-art accuracy in computer vision (Hénaff et al., 2019; Gidaris et al., 2019) and natural language processing (Devlin et al., 2018) supervised learning tasks. The use of self-supervised learning in graph representation learning is intimately related to the use of pseudolikelihood to approximate true likelihood functions.

For further related work on collective classification, see Appendix H.

7. Conclusion

A long-standing question is when/if collective inference (CI) is needed when very expressive graph models are available (e.g., GNNs) for inductive node classification tasks. This work solves a few theoretical and empirical questions towards an answer. We show that, with the most expressive equivariant (node-embedding) GNNs, it is true that there is no need for collective learning. While the development of more expressive GNNs generally focuses on changing the architecture, in this work we ask the question of whether CI could be a practical way to boost the real-world performance of a GNN, without changing its underlying architecture.

In this work we propose *collective learning* (CL), a modified CI approach for GNN-type classifiers that boosts their expressiveness, relying on both Monte Carlo sampling of node embeddings and (self-supervised) random masking in training. We show that collective learning can be combined with existing GNNs to improve their expressiveness (and we prove increased expressiveness with WL-GNNs).

We experimentally confirm our theoretical analysis across five real-world graphs and four component GNNs, and show by extensive empirical study that *CL+GNN* consistently, and significantly, boosts GNNs performance (up to 26%). One limitation of our proposed collective learning framework is the computational cost of using sampled embeddings during each stochastic gradient step. We leave exploration of mechanisms to reduce the additional computational burden (eg. via parallelization and/or more targeted sampling) to future work.

8. Acknowledgments

This research is supported by NSF under contract number(s) CAREER IIS-1943364, IIS-1618690, CCF-0939370 and CCF-1918483. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of NSF or the U.S. Government.

References

Azizian, W. and Lelarge, M. Characterizing the expressive power of invariant and equivariant graph neural networks. 2021.

Bakshy, E., Eckles, D., and Bernstein, M. S. Designing and deploying online field experiments. In *Proceedings of the 23rd international conference on World wide web*, pp. 283–292, 2014.

Beaini, D., Passaro, S., L'etourneau, V., Hamilton, W. L., Corso, G., and Li'o, P. Directional graph networks. *arXiv preprint arXiv:2010.02863*, 2020.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., and Kriegel, H.-P. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005.

Bouritsas, G., Frasca, F., Zafeiriou, S., and Bronstein, M. M. Improving graph neural network expressivity via subgraph isomorphism counting. *arXiv preprint arXiv:2006.09252*, 2020.

Chen, Z., Villar, S., Chen, L., and Bruna, J. On the equivalence between graph isomorphism testing and function approximation with gnns. In *Advances in Neural Information Processing Systems*, pp. 15868–15876, 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Doersch, C., Gupta, A., and Efros, A. A. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1422–1430, 2015.

Fan, S. and Huang, B. Recurrent collective classification. *Knowledge and Information Systems*, 60(2):741–755, 2019.

Feng, W., Zhang, J., Dong, Y., Han, Y., Luan, H., Xu, Q., Yang, Q., Kharlamov, E., and Tang, J. Graph random neural networks for semi-supervised learning on graphs. *Advances in Neural Information Processing Systems*, 33, 2020.

Gidaris, S., Bursuc, A., Komodakis, N., Pérez, P., and Cord, M. Boosting few-shot visual learning with self-supervision. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 8059–8068, 2019.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

Hénaff, O. J., Razavi, A., Doersch, C., Eslami, S., and Oord, A. v. d. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019.

Jensen, D., Neville, J., and Gallagher, B. Why collective inference improves relational classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 593–598, 2004.

Jia, J. and Benson, A. Outcome correlation in graph neural network regression, 2020.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Koller, D., Friedman, N., Džeroski, S., Sutton, C., McCallum, A., Pfeffer, A., Abbeel, P., Wong, M.-F., Heckerman, D., Meek, C., et al. *Introduction to statistical relational learning*. MIT press, 2007.

Lee, H.-Y., Huang, J.-B., Singh, M., and Yang, M.-H. Unsupervised representation learning by sorting sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 667–676, 2017.

Lu, Q. and Getoor, L. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 496–503, 2003.

Luan, S., Zhao, M., Chang, X.-W., and Precup, D. Break the ceiling: Stronger multi-scale deep graph convolutional networks. *arXiv preprint arXiv:1906.02174*, 2019.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. Provably powerful graph networks. In *Advances in neural information processing systems*, pp. 2156–2167, 2019a.

Maron, H., Fetaya, E., Segol, N., and Lipman, Y. On the universality of invariant networks. In *International conference on machine learning*, pp. 4363–4371. PMLR, 2019b.

Misra, I., Zitnick, C. L., and Hebert, M. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pp. 527–544. Springer, 2016.

Moore, J. and Neville, J. Deep collective inference. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

Murphy, R. L., Srinivasan, B., Rao, V., and Ribeiro, B. Relational pooling for graph representations. *arXiv preprint arXiv:1903.02541*, 2019.

Neville, J. and Jensen, D. Iterative classification in relational data. In *Proc. AAAI-2000 workshop on learning statistical models from relational data*, pp. 13–20, 2000.

Neville, J., Jensen, D., Friedland, L., and Hay, M. Learning relational probability trees. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 625–630, 2003a.

Neville, J., Jensen, D., and Gallagher, B. Simple estimators for relational bayesian classifiers. In *Third IEEE International Conference on Data Mining*, pp. 609–612. IEEE, 2003b.

Noroozi, M. and Favaro, P. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pp. 69–84. Springer, 2016.

Pfeiffer III, J. J., Neville, J., and Bennett, P. N. Overcoming relational learning biases to accurately predict preferences in large scale networks. In *Proceedings of the 24th International Conference on World Wide Web*, pp. 853–863, 2015.

Pham, T., Tran, T., Phung, D., and Venkatesh, S. Column networks for collective classification. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Popescul, A., Ungar, L. H., Lawrence, S., and Pennock, D. M. Towards structural logistic regression: Combining relational and statistical learning. *Departmental Papers (CIS)*, pp. 134, 2002.

Qu, M., Bengio, Y., and Tang, J. Gmnn: Graph markov neural networks. *arXiv preprint arXiv:1905.06214*, 2019.

Robbins, H. and Monro, S. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 09 1951. doi: 10.1214/aoms/1177729586. URL https://doi.org/10.1214/aoms/1177729586.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Srinivasan, B. and Ribeiro, B. On the equivalence between node embeddings and structural graph representations. *arXiv preprint arXiv:1910.00452*, 2019.

Teixeira, L., Jalaian, B., and Ribeiro, B. Are graph neural networks miscalibrated? *arXiv preprint arXiv:1905.02296*, 2019.

Vignac, C., Loukas, A., and Frossard, P. Building powerful and equivariant graph neural networks with structural message-passing. *arXiv e-prints*, pp. arXiv–2006, 2020.

Vijayan, P., Chandak, Y., Khapra, M. M., Parthasarathy, S., and Ravindran, B. Hopf: Higher order propagation framework for deep collective classification. *arXiv preprint arXiv:1805.12421*, 2018.

Xiang, R. and Neville, J. Pseudolikelihood em for within-network relational learning. In *2008 Eighth IEEE International Conference on Data Mining*, pp. 1103–1108. IEEE, 2008.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Yang, J., Ribeiro, B., and Neville, J. Stochastic gradient descent for relational logistic regression via partial network crawls. *arXiv preprint arXiv:1707.07716*, 2017.