COMBINING LABEL PROPAGATION AND SIMPLE MODELS OUT-PERFORMS GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Neural Networks (GNNs) are a predominant technique for learning over graphs. However, there has been relatively little effort put into understanding why GNNs are successful and whether they are necessary for good performance. Here, we show that for many standard transductive node classification benchmarks, we can exceed or match the performance of state-of-the-art GNNs by combining shallow models that ignore the graph structure entirely with two simple postprocessing steps that exploit correlation in the label structure: (i) an "error correlation" that spreads residual errors in training data to correct errors in test data and (ii) a "prediction correlation" that smooths the predictions on the test data. Using these correlations, which we call Correct and Smooth (C&S), are implemented via simple modifications to standard label propagation techniques from early graph-based semi-supervised learning methods. Our approach exceeds or nearly matches the performance of state-of-the-art GNNs on a wide variety of benchmarks, with just a small fraction of the parameters and orders of magnitude faster runtime. For instance, we exceed the best known GNN performance on the OGB-Products dataset with 137x fewer parameters and >100x less training time. The performance of our methods highlights how directly incorporating label information into the learning algorithm (as was done in traditional techniques) yields easy and substantial performance gains. We can also incorporate our techniques into big GNN models, providing modest gains.

1 INTRODUCTION

Following the success of neural networks in computer vision and natural language processing, there are now a wide range of *graph neural networks* (GNNs) for making predictions involving relational data (Battaglia et al., 2018; Wu et al., 2020). These models have had much success and sit atop leaderboards such as the Open Graph Benchmark (Hu et al., 2020). Often, the methodological developments for GNNs have revolved around creating strictly more expressive architectures than basic variants such as the Graph Convolutional Network (GCN) (Kipf & Welling, 2017) or GraphSAGE (Hamilton et al., 2017a); examples include Graph Attention Networks (Veličković et al., 2018), Graph Isomorphism Networks (Xu et al., 2018), and various deep models (Li et al., 2019; Rong et al., 2019; Chen et al., 2020). Many of these ideas are adapted from new architectures in models for vision (e.g., deep CNNs) or language (e.g., attention), with the hopes that success will translate to graphs. However, as these models become more complex, understanding their performance gains is a major challenge, and scaling them to large datasets is difficult.

Here, we see how far we can get by combining much simpler models, with an emphasis on better understanding where there are easy opportunities for performance improvements in graph learning, particularly transductive node classification. We propose a simple pipeline with three main parts (Fig. 1): (i) a base prediction made with node features that ignores the graph structure (e.g., an MLP or linear model); (ii) a correction step, which propagates uncertainties from the training data across the graph to correct the base prediction; and (iii) a smoothing of the predictions over the graph. Steps (ii) and (iii) are just post-processing and use classical methods for graph-based semi-supervised learning — namely, label propagation (Zhu, 2005). With a few modifications and new deployment of these classic ideas, we achieve state-of-the-art performance on several node classification tasks, outperforming big GNN models. In our framework, the graph structure is not used to learn parameters but instead as a pre and post-processing mechanism. This simplicity leads to

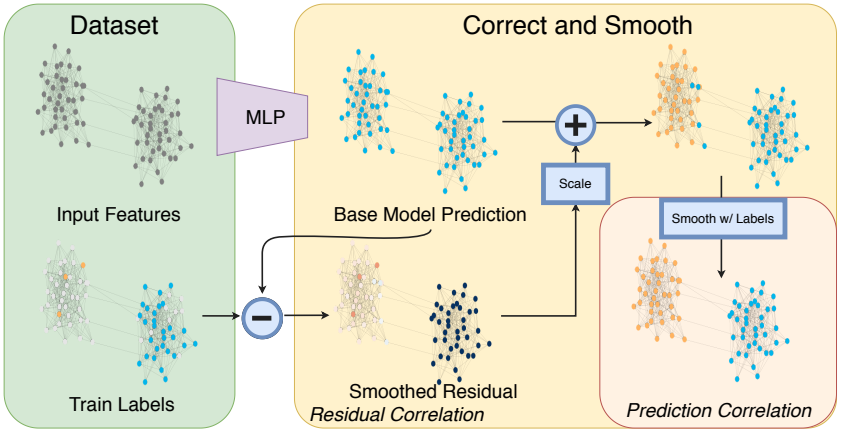


Figure 1: Overview of our GNN-free model, Correct and Smooth, with a toy example. The left cluster belongs to orange and the right cluster belongs to blue. We use MLPs for base predictions, ignoring the graph structure, which we assume gives the same prediction on all nodes in this example. After, base predictions are corrected by propagating errors from the training data. Finally, corrected predictions are smoothed with label propagation.

models with orders of magnitude fewer parameters that take orders of magnitude less time to train, and can easily scale to massive graphs. We can also combine our ideas with state-of-the-art GNNs, and see modest performance gains.

A major source of our performance improvements is directly using labels for predictions. This idea is not new — early diffusion-based semi-supervised learning algorithms on graphs such as label propagation (Zhu et al., 2003), label spreading (Zhou et al., 2004), and the spectral graph transducer (Joachims, 2003) all use this idea. However, the motivation for these methods was semi-supervised learning on point cloud data, so the features were used to construct the graph. Since then, these techniques have been used for learning on relational data from just the labels (i.e., no features) (Koutra et al., 2011; Gleich & Mahoney, 2015; Peel, 2017; Chin et al., 2019) but have largely been ignored in GNNs. However, we see that simple label propagation (which ignores features) does surprisingly well on a number of benchmarks. This provides motivation for combining two orthogonal sources of prediction power — one coming from the node features (ignoring graph structure) and one coming from using the known labels directly in predictions.

Recent research connects GNNs to label propagation (Wang & Leskovec, 2020; Jia & Benson, 2020), Markov Random fields (Qu et al., 2019; Gao et al., 2019) and has also included ad hoc incorporation of label information in the features (Shi et al., 2020). However, these approaches are still expensive to train, while we use label propagation in two understandable and low-cost ways. We start with a cheap "base prediction" from a model that ignores graph structure (apart from perhaps pre-processed features like spectral embeddings). After, we first use label propagation for error correction and then to smooth final predictions. These post-processing steps are based on the fact that errors and labels on connected nodes are positively correlated. Assuming similarity between connected nodes is at the center of much network analysis and corresponds to homophily or assortative mixing (McPherson et al., 2001; Newman, 2003; Easley & Kleinberg, 2010). In the semi-supervised learning learning literature, the analog is the smoothness or cluster assumption (Chapelle et al., 2003; Zhu, 2005). The surprisingly solid performance of label propagation across a wide variety of datasets suggests that these assumptions hold true across common benchmarks.

Overall, our methodology demonstrates that combining several simple ideas yields excellent performance in transductive node classification at a fraction of the cost, in terms of both model size (i.e., number of parameters) and training time. For example, on the OGB-Products benchmark, we out-perform the current best-known GNN with $>$100x fewer parameters and $>$100x less training time. However, our goal is *not* to say that current graph learning methods are poor or inappropriate. Instead, we aim to highlight easier ways in which to improve prediction performance in graph learning. More specifically, we find that more direct incorporation of labels into the learning algorithms is key. And by combining our ideas with existing GNNs, we also see improvements (albeit modest ones). We hope that our approach spurs new ideas that can help in other graph learning tasks, such as inductive node classification, link prediction, and graph prediction.

1.1 ADDITIONAL RELATED WORK

There are several recent approaches that incorporate nonlinearity into label propagation methods to compete with GNNs and achieve scalability (Eliav & Cohen, 2018; Ibrahim & Gleich, 2019; Tudisco et al., 2020), but these methods focus on settings of low label rates. The Approximate Personalized Propagation of Neural Predictions (APPNP) framework is most relevant to our work, as they also smooth base predictions (Klicpera et al., 2018). However, they focus on integrating this smoothing into the training process so their model can be trained end to end. Not only is this significantly more computationally expensive, it also prevents APPNP from incorporating label information at inference. Compared to APPNP, our framework not only performs significantly better, it also runs faster and scales much easier. Our framework also complements the Simplified Graph Convolution (Wu et al., 2019), as well as algorithms designed to increase scalability (Bojchevski et al., 2020; Zeng et al., 2019; Rossi et al., 2020). The primary focus of our approach, however, is using labels directly, and scalability is a byproduct. Finally, there is also prior work connecting GCNs and label propagation. Wang & Leskovec (2020) use label propagation as a pre-processing step to weight edges, whereas we use label propagation as a post-processing step. Jia & Benson (2020) use label propagation with GNNs for regression tasks, and our error propagation adapts some of the ideas for the case of classification.

2 CORRECT AND SMOOTH MODEL

We start with some notation. Suppose we have an undirected graph $G = (V, E)$, where there are $n = |V|$ nodes with features on each node represented by a matrix $X \in \mathbb{R}^{n \times p}$. Let A be the adjacency matrix of the graph, D be the diagonal degree matrix, and S be the normalized adjacency matrix $D^{-1/2}AD^{-1/2}$. For the prediction problem, the node set V is split into a disjoint set of unlabeled nodes U and labeled nodes L, which are subsets of the indices $\{1, \ldots, n\}$. We represent the labels by a one-hot-encoding matrix $Y \in \mathbb{R}^{n \times C}$, where C is the number of classes (i.e., $Y_{ic} = 1$ if $i \in L$ is in class c, and 0 otherwise), and we further split the labeled nodes into a training set L_t and validation set L_v. Our problem is transductive node classification: assign each node $j \in U$ a label in $\{1, \ldots, C\}$, given G, X, and Y.

Our approach starts with a simple base predictor on node features, which does not rely on any learning over the graph. After, we perform two types of label propagation (LP): one that corrects the base predictions by modeling correlated error and one that smooths the final prediction. We call the combination of these two methods Correct and Smooth (C&S; Fig. 1). The LPs are only post-processing steps — our pipeline is *not* trained end-to-end. Furthermore, the graph is only used in these post-processing steps and the spectral embedding, but not for the base predictions. This makes training fast and scalable compared to standard GNN models. Moreover, we take advantage of both LP (which tends to perform fairly well on its own without features) and the node features. We will see that combining these complementary signals yields excellent predictions.

2.1 SIMPLE BASE PREDICTOR

To start, we use a simple base predictor that does not rely on the graph structure. More specifically, we train a model f to minimize $\sum_{i \in L_t} \ell(f(x_i), y_i)$, where x_i is the ith row of X, y_i is the ith row of Y, ℓ is a loss function. For this paper, f is either a linear model or a shallow MLP followed by a softmax, and ℓ is the cross-entropy loss. The validation set L_v is used to tune hyperparameters such as weight decay and the hidden layer dimensions for the MLP. From f, we get a *base prediction* $Z \in \mathbb{R}^{n \times c}$, where each row of Z is a probability distribution resulting from the softmax. Omitting the graph structure for these base predictions avoids the scalability issues with GNNs. In principle, though, we can use any base predictor for Z, including those based on GNNs, and we explore this in Section 3. However, for our pipeline to be simple and scalable, we just use linear classifiers or MLPs with subsequent post-processing, which we describe next.

2.2 CORRECTING FOR ERROR IN BASE PREDICTIONS WITH RESIDUAL PROPAGATION

Next, we improve the accuracy of Z by incorporating labels to correct errors in the base prediction. The key idea is that we expect *errors* in the base prediction to be correlated according to the graph. In

other words, an error at node i increases the chance of an error at neighboring nodes of i and should spread uncertainty over the graph. Our approach is inspired in part by the residual propagation (Jia & Benson, 2020), where a similar concept is used for node regression tasks, as well as generalized least squares and correlated error models more broadly (Shalizi, 2013).

To this end, we first define an error matrix $E \in \mathbb{R}^{n \times c}$, where error is the *residual* on the training data and zero elsewhere:

$$E_{L_t} = Z_{L_t} - Y_{L_t}, \quad E_{L_v} = 0, \quad E_U = 0. \tag{1}$$

Note that $E = 0$ only when the MLP makes perfect predictions on the training nodes. We then smooth this error using the label spreading technique of Zhou et al. (2004), optimizing the objective

$$\hat{E} = \underset{W \in \mathbb{R}^{n \times c}}{\arg\min} \operatorname{trace}(W^T S W) + \mu \|W - E\|_F^2. \tag{2}$$

The first term encourages smoothness of the error estimation over the graph, and the second term keeps estimations close to our initial guess of the error E. As in Zhou et al. (2004), the solution can be obtained via the iteration $E^{(t+1)} = (1-\alpha)E + \alpha S E^{(t)}$, where $\alpha = 1/(1+\mu)$ and $E^{(0)} = E$, which converges rapidly to $\hat{E}$. This iteration can be thought of as a spreading, diffusion, or propagation of the error. Finally, we add the errors back to the base prediction, forming the correlated predictions $Z^{(r)} = Z + \hat{E}$. We emphasize that this is a post-processing technique and there is no coupled training with the base predictions.

Under a Gaussian assumption in regression, this propagation suffices (Jia & Benson, 2020). However, for the classification problems we consider, the smoothed errors $\hat{E}$ might not be at the right scale. In particular, $||\hat{E}||_1 \leq ||E||_1$, which is less than $||Y - Z||_1$ (i.e: the residual needed to completely correct all of the model's predictions). This means that it is impossible for the above propagation to completely correct all nodes in the graph – it simply does not have enough "mass". To address this, we propose two variations.

Autoscale. Intuitively, we want to scale the size of errors in $\hat{E}$ to be approximately the size of the errors in E. We only know the true errors at labeled nodes, so we approximate the scale with the average error over the training nodes. Formally, let $e_j \in \mathbb{R}^c$ correspond to the jth row of E, and define $\sigma = \frac{1}{|L_t|} \sum_{j \in L_t} \|e_j\|_1$. Then the corrected predictions on an unlabeled node i is given by $Z^{(r)}_{i,:} = Z_{i,:} + \sigma \hat{E}_{:,i} / \|\hat{E}^T_{:,i}\|_1$ for $i \in U$.

Scaled Fixed Diffusion (FDiff-scale). Alternatively, we can use a diffusion like the one from Zhu et al. (2003), which keeps the known errors at training nodes *fixed*. More specifically, we iterate $E^{(t+1)}_U = [D^{-1} A E^{(t)}]_U$ and keep fixed $E^{(t)}_L = E_L$ until convergence to $\hat{E}$, starting with $E^{(0)} = E$. Intuitively, this fixes error values where we know the error (on the labeled nodes L), while other nodes keep averaging over the values of their neighbors until convergence. With this type of propagation, the maximum and minimum values of entries in $E^{(t)}$ do not go beyond those in E_L. We still find it effective to introduce a scaling hyperparameter s so that $Z^{(r)} = Z + s\hat{E}$.

2.3 Smoothing final predictions with Prediction Correlation

At this point, we have a score vector $Z^{(r)}$, obtained from correcting the base predictor Z with a model for the correlated error $\hat{E}$. To make a final prediction, we further smooth the correct prediction. The motivation is that adjacent nodes in the graph are likely to have similar labels, which is common given homophily or assortative properties of a network. Thus, we can encourage smoothness over the distribution over labels by propagation again (Zhou et al., 2004). First, we start with our best guess $G \in \mathbb{R}^{n \times c}$ of the labels:

$$G_{L_t} = Y_{L_t}, \quad G_{L_v, U} = Z^{(r)}_{L_v, U}. \tag{3}$$

Here, we set the training nodes back to their true labels and use the corrected predictions for the validation and unlabeled nodes (we can also use the true validation labels, which we discuss later in the experiments). We then iterate $G^{(t+1)} = (1-\alpha)G + \alpha S G^{(t)}$ with $G^{(0)} = G$ until convergence to give the final prediction $\hat{Y}$. The classification for a node $i \in U$ is $\arg\max_c \hat{Y}_{ic}$.

As with error correlation, the smoothing here is a post-processing step, decoupled from the other steps. This type of prediction smoothing is similar in spirit to APPNP (Klicpera et al., 2018) (which we compare against later). However, APPNP is trained end-to-end, propagates on final-layer representations instead of softmaxes, does not use labels, and is motivated differently.

2.4 Summary and additional considerations

To review our pipeline, we start with a cheap base prediction Z, using only node features but not the graph structure. After, we estimate errors $\hat{E}$ by propagating known errors on the training data, resulting in error-corrected predictions $Z^{(r)} = Z + \hat{E}$. Finally, we treat these as score vectors on unlabeled nodes, and combine them with the known labels through another LP step to produce smoothed final predictions. We refer to this general pipeline as **Correct and Smooth** (C&S).

Before showing that this pipeline achieves state-of-the-art performance on transductive node classification, we briefly describe another simple way of improving performance: feature augmentation. The hallmark of deep learning is that we can learn features instead of engineering them. However, GNNs still rely on informative input features to make predictions. There are numerous ways to get useful features from just the graph topology to augment the raw node features (Henderson et al., 2011; 2012; Hamilton et al., 2017b). In our pipeline, we augment features with a regularized spectral embedding (Chaudhuri et al., 2012; Zhang & Rohe, 2018) coming from the leading k eigenvectors of the matrix $D_\tau^{-1/2}(A + \frac{\tau}{n}\mathbf{1}\mathbf{1}^T)D_\tau^{-1/2}$, where $\mathbf{1}$ is a vector of all ones, τ is a regularization parameter set to the average degree, and D_τ is diagonal with ith diagonal entry equals $D_{ii} + \tau$. The underlying matrix is dense, but we can apply matrix-vector products in linear time and use iterative eigensolvers to compute the embedding efficiently (for a sense of runtime, it takes about 15 seconds to compute a 128-dimensional spectral embedding on graphs with a million edges).

3 Experiments on Transductive Node Classification

To demonstrate the effectiveness of our methods, we use 9 datasets (Table 1). The Arxiv and Products datasets are from the Open Graph Benchmark (OGB) (Hu et al., 2020); the Cora, Citeseer, and Pubmed are three classic citation network benchmarks (Getoor et al., 2001; Getoor, 2005; Namata et al., 2012); and wikiCS is a web graph (Mernyei & Cangea, 2020). In these datasets, classes are categories of papers, products, or pages , and features are derived from text. We also use a Facebook social network of Rice University, where classes are dorm residences and features are attributes such as gender, major, and class year, amongst others (Traud et al., 2012), as well as a geographic dataset of US counties where classes are 2016 election outcomes and features are demographic (Jia & Benson, 2020). Finally, we use an email dataset of a European research institute, where classes are department membership and there are no features (Leskovec et al., 2007; Yin et al., 2017).

Data splits. The training/validation/test splits for Arxiv and Products are given by the benchmark, and the splits for wikiCS come from Mernyei & Cangea (2020). For the Rice, US counties, and email data, we use 40%/10%/50% random splits, and for the smaller citation networks, we use 60%/20%/20% random splits, as in Wang & Leskovec (2020) (in contrast to lower label rate settings (Yang et al., 2016)) to ameliorate sensitivity to hyperparameters.

Table 1: Summary statistics of datasets. We include the reduction in the number of parameters and change in accuracy compared to SOTA for our best C&S model, along with training time.

Datasets	Classes	Nodes	Edges	Parameter Δ	Accuracy Δ	Time
Arxiv	40	169,343	1,166,243	-84.9%	+ 0.97	9.89 s
Products	47	2,449,029	61,859,140	-93.47%	+1.53	170.6 s
Cora	7	2,708	5,429	-98.37%	+ 1.09	0.5 s
Citeseer	6	3,327	4,732	-89.68%	-0.69	0.48 s
Pubmed	3	19,717	44,338	-96.00%	- 0.30	0.85 s
Email	42	1,005	25,571	- 97.89%	+ 4.26	42.83 s
Rice31	10	4,087	184,828	- 99.02%	+ 1.39	39.33 s
US County	2	3,234	12,717	- 74.56%	+ 1.77	39.05 s
wikiCS	10	11,701	216,123	- 84.88%	+ 2.03	7.09 s

Base predictors and other models. We use *Linear* and *MLP* models as simple base predictors, where the input features are the raw node features and the spectral embedding. We also use a *Plain Linear* model that only uses the raw features for comparison and Label Propagation (*LP*; specifically, the Zhou et al. (2004) version), which only uses labels. For comparable GNN models to our framework (in terms of simplicity or style), we use GCN, SGC, and APPNP. Finally, we include several "state-of-the-art" (SOTA) baselines. For Arxiv and Products, this is UniMP (Shi et al., 2020) (top of OGB leaderboard, as of October 1, 2020). For For Cora, Citeseer and Pubmed, we reuse the top performance scores from Chen et al. (2020). For Email and US County, we use GCNII (Chen et al., 2020). For Rice31 we use GCN with spectral and node2vec (Grover & Leskovec, 2016) embeddings. For wikiCS, we use APPNP as reported by Mernyei & Cangea (2020). We select a set of fixed hyperparameters using the validation set. See the appendix for more details.

3.1 Results on node classification

In our first set of results, we only use the training labels in our C&S framework. For the results discussed here, this is generous to our baselines. The ability to include validation labels is an advantage of our approach, and this improves performance of our framework even further (Table 1). We discuss this in the next section.

Table 2 reports results. We highlight a few important findings. First, within our model, there are substantial gains from the LP post-processing steps (for instance, on Products, the MLP base prediction goes from 74% to 83%). Second, even Plain Linear with C&S is sufficient to outperform plain

Table 2: Performance of our C&S framework, using only the ground truth training labels in Eq. 3. Further improvements can be made by including ground truth validation labels (see Table 1).

Methods	Base Model	Arxiv	Products	Cora	Citeseer	Pubmed
LP	—	68.5	74.76	86.50	70.64	83.74
Plain GCN	—	71.74	75.64	85.77	73.68	88.13
SGC	Plain Linear	69.39	68.83	86.81	72.04	84.04
APPNP	Plain Linear	66.38	OOM	87.87	76.53	89.40
SOTA	—	**73.79**	82.56	88.49	**77.99**	**90.30**
Base Prediction	Plain Linear	52.32	47.70	73.85	70.27	87.10
	Linear	70.08	72.08	74.75	70.51	87.19
	MLP	71.51	73.97	74.06	68.10	86.85
Autoscale	Plain Linear	71.11	80.24	88.62	76.31	89.99
	Linear	72.07	80.25	**88.73**	76.75	89.93
	MLP	72.62	78.60	87.39	76.31	89.33
FDiff-scale	Plain Linear	70.60	81.84	89.05	76.22	89.74
	Linear	71.57	82.65	88.66	77.06	89.51
	MLP	72.43	**83.58**	87.39	76.42	89.23
Methods	**Base Model**	**Email**	**Rice31**	**US County**	**wikiCS**	
LP	—	70.69	82.19	87.90	76.72	
Plain GCN	—	—	15.45	84.13	78.61	
SGC	Plain Linear	—	16.59	83.92	72.86	
APPNP	Plain Linear	70.28	11.34	84.14	69.83	
SOTA	—	71.96	86.50	88.08	**79.84**	
Base Prediction	Plain Linear	—	9.84	75.74	72.45	
	Linear	66.24	70.26	84.07	74.29	
	MLP	69.13	17.16	87.70	73.07	
Autoscale	Plain Linear	—	75.99	85.25	79.57	
	Linear	72.50	86.42	86.15	79.53	
	MLP	74.55	85.50	89.64	78.10	
FDiff-scale	Plain Linear	—	73.66	87.38	79.54	
	Linear	72.53	**87.55**	88.11	79.25	
	MLP	**75.74**	85.74	**89.85**	78.24	

GCNs in many cases, and LP is often competitive with GCNs. This is striking given that the main motivation for GCNs was to address the fact that connected nodes may not have similar labels (Kipf & Welling, 2017). Our results suggest that directly incorporating correlation in the graph with simple use of the features is often a better idea. Third, our model variants can out-perform SOTA on Products, Cora, Email, Rice31, and US County (often substantially so); on the other datasets, there is not much difference between our best-performing model and the SOTA. Next, we see how to improve performance of C&S even further.

3.2 FURTHER IMPROVEMENTS BY USING MORE LABELS

To get a sense of how much using ground truth labels directly helps, we also experiment with a version of C&S *without labels*. Instead of running our LP steps, we just smooth the output of the base predictors using the approach of Zhou et al. (2004) (Table 3) and call this the *Basic Model*. We see that the linear and MLP base predictor can often exceed the performance of a GCN. Again, this suggests that smoothed outputs are important, and that the original motivations for GCNs are perhaps misleading. Instead, we hypothesize that GCNs gain performance by having smoothed outputs over the graph, a similar observation with Wu et al. (2019). However, there are still gaps in performance between our models here and those in the previous section that directly used labels.

Table 3: Basic Model performance.

Base Model	Arxiv	Products
Plain Linear	63.30	66.27
Linear	71.42	78.73
MLP	**72.48**	**80.34**
Plain GCN	71.74	75.64

Table 4: C&S with train and validation labels. The standard deviation in all results is < 0.3, except for GCN, which has standard deviation < 0.5.

Methods	Base Model	Arxiv	Products
Autoscale	Plain Linear	72.71	80.55
	Linear	73.78	80.56
	MLP	**74.02**	79.29
FDiff-scale	Plain Linear	72.42	82.71
	Linear	72.93	82.81
	MLP	73.46	**84.55**
SOTA	—	73.65	82.56

We improve C&S performance further by using both training and validation labels in Eq. 3 instead of just the training labels. (We do *not* use validation labels to update the base prediction model after training — they are just used to select hyperparameters) Using validation labels boosts performance even further (partial results in Table 4; Table 1 shows gains over SOTA. See appendix for more results). The ability to incorporate labels is a benefit of our approach. On the other hand, GNNs do not have this advantage, as they often rely on early stopping to prevent overfitting, may not always benefit from more data (e.g., under distributional shift in the validation and test set), and do not directly use labels. Thus, our comparisons in Table 2 are more generous than needed. With validation labels, our best model out-performs SOTA improves the performance of our model in all cases, often by substantial margins (Table 1).

We note that the evaluation procedure for GNN benchmarks differ from those for classical LP. For GNNs, a sizable validation set is often used (and needed) for substantial hyperparameter tuning, as well as early stopping. With LP, one can use the entire set of labeled nodes L with cross-validation to select the single hyperparameter α. Given the setup of transductive node classification, however, there is no reason not to use validation labels at inference if they are helpful (e.g., via LP in our case).

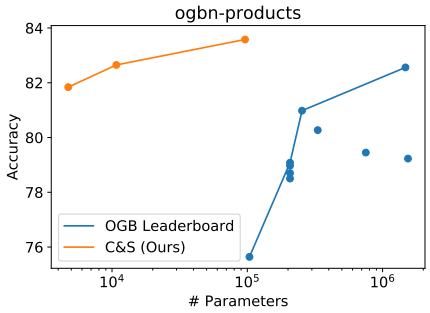
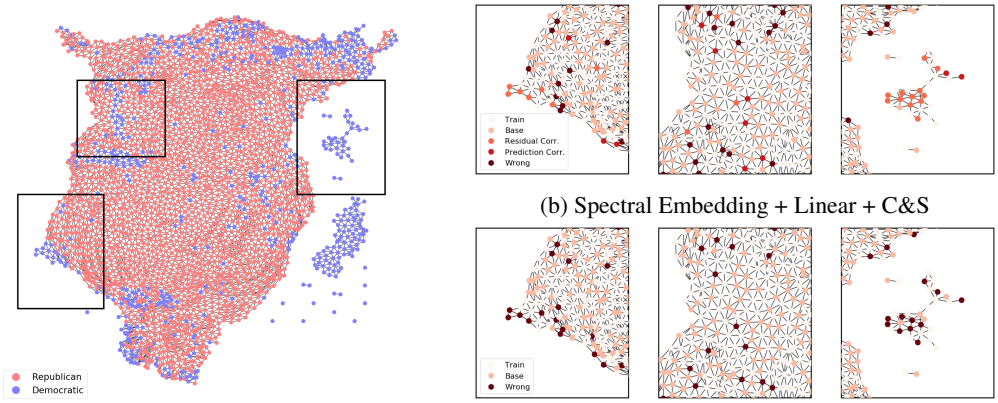


Figure 2: Accuracy and model size on Products.

Table 5: C&S with GNN base predictors.

Dataset	Model	Performance
ogbn-arxiv	GAT	73.52 ± 0.02
	GAT + C&S	**73.83** ± 0.03
	SOTA	73.79 ± 0.25
US County	GCNII (SOTA)	88.08 ± 2.15
	GCNII + C&S	**89.59** ± 1.06



(a) Ground Truth

(b) Spectral Embedding + Linear + C&S

(c) Spectral Embedding + GCN

Figure 3: US County visualizations of which stage of C&S got each node correct.

3.3 Faster training and improving existing GNNs

C&S often requires significantly fewer parameters compared to GNNs or other SOTA solutions. As an example, we plot parameters vs. performance for Products in Fig. 2. While having fewer parameters can be useful, the real gain is in faster training time, and our models are typically orders of magnitude faster to train than models with comparable accuracy because we do not use the graph structure for our base predictions. As one example, although our MLP + C&S model based for the Arxiv dataset has a similar number of parameters compared to the GCN + labels method on the OGB leaderboards, our model runs 7x faster per epoch and converges much faster. In addition, compared to the SOTA for the Products dataset, *our framework with a linear base predictor has higher accuracy, trains* $> 100x$ *faster, and has 137x fewer parameters*. Finally, we evaluate on OGB papers100M, the largest node classification benchmark available, and obtain 65.33%, beating the SOTA (63.29%).

Our pipeline can also be used to improve the performance of GNNs in general. We applied our error correction and final prediction smoothing to more complex base predictors such as GCNII or GAT. This improves our results on some datasets, including beating SOTA on ogbn-arxiv (Table 5). However, the performance improvements are sometimes only minor, suggesting that big models might be capturing the same signal as our simple C&S framework.

3.4 Qualitative analysis

To aid in understanding the performance of C&S, we visualize the predictions on the US County dataset (Fig. 3). We see that the residual correlation tends to correct nodes with very correlated errors. For example, in the lower left (Texas) and upper right (Hawaii) of the Us County graph, corresponding to the left and right crops in Figs. 3b and 3c, many errors of the base linear model are corrected by the residual correlation. This is because the demographic features are misleading in those counties, leading both the linear model and GCN to get it wrong. However, only the residual correlation is able to fix them. Finally, we observe the final prediction correlation will smooth the prediction, as shown in the center crop, so that the errors can be fixed based on the correct classification of the neighbors. We observe similar behavior on Rice31 (see the appendix).

4 Conclusion

GNN models are becoming more expressive, more parameterized, and more expensive to train. Our results suggest that, at least on transductive node classification tasks, it is worth exploring other types of techniques for improving performance. This can lead to both better performance and much faster training. In particular, label propagation and its variants are longstanding powerful tools, and more directly incorporating them into graph learning models has major benefits.

REFERENCES

P. Battaglia, Jessica B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, Mateusz Malinowski, Andrea Tacchetti, D. Raposo, Adam Santoro, R. Faulkner, Çaglar Gülçehre, H. Song, A. Ballard, J. Gilmer, G. Dahl, Ashish Vaswani, Kelsey R. Allen, C. Nash, V. Langston, Chris Dyer, N. Heess, Daan Wierstra, Pushmeet Kohli, M. Botvinick, Oriol Vinyals, Y. Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *ArXiv*, abs/1806.01261, 2018.

Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2464–2473, 2020.

Olivier Chapelle, Jason Weston, and Bernhard Schölkopf. Cluster kernels for semi-supervised learning. In *Advances in neural information processing systems*, pp. 601–608, 2003.

Kamalika Chaudhuri, Fan Chung, and Alexander Tsiatas. Spectral clustering of graphs with general degrees in the extended planted partition model. In *Conference on Learning Theory*, pp. 35–1, 2012.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *Proceedings of the International Conference on Machine Learning*, 2020.

Alex Chin, Yatong Chen, Kristen M. Altenburger, and Johan Ugander. Decoupled smoothing on graphs. In *The World Wide Web Conference*, pp. 263–272, 2019.

David Easley and Jon Kleinberg. *Networks, crowds, and markets*. Cambridge University Press, 2010.

Buchnik Eliav and Edith Cohen. Bootstrapped graph diffusions: Exposing the power of nonlinearity. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2(1):1–19, 2018.

Hongchang Gao, Jian Pei, and Heng Huang. Conditional random field enhanced graph convolutional neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 276–284, 2019.

Lise Getoor. Link-based classification. In *Advanced methods for knowledge discovery from complex data*, pp. 189–207. Springer, 2005.

Lise Getoor, Nir Friedman, Daphne Koller, and Benjamin Taskar. Learning probabilistic models of relational structure. In *ICML*, volume 1, pp. 170–177. Citeseer, 2001.

David F Gleich and Michael W Mahoney. Using local spectral methods to robustify graph-based learning algorithms. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 359–368, 2015.

Aditya Grover and J. Leskovec. node2vec: Scalable feature learning for networks. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017a.

William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 2017b.

Keith Henderson, Brian Gallagher, Lei Li, Leman Akoglu, Tina Eliassi-Rad, Hanghang Tong, and Christos Faloutsos. It's who you know: graph mining using recursive structural features. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 663–671, 2011.

Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. RolX: structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1231–1239, 2012.

Weihua Hu, M. Fey, M. Zitnik, Yuxiao Dong, H. Ren, Bowen Liu, Michele Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *ArXiv*, abs/2005.00687, 2020.

Rania Ibrahim and David Gleich. Nonlinear diffusion for community detection and semi-supervised learning. In *The World Wide Web Conference*, pp. 739–750, 2019.

Junteng Jia and Austin R. Benson. Residual correlation in graph neural network regression. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 2020.

Thorsten Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the International Conference on Machine Learning*, pp. 290–297, 2003.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2018.

Danai Koutra, Tai-You Ke, U Kang, Duen Horng Polo Chau, Hsing-Kuo Kenneth Pao, and Christos Faloutsos. Unifying guilt-by-association approaches: Theorems and fast algorithms. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 245–260. Springer, 2011.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2–es, 2007.

Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

P'eter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *ArXiv*, abs/2007.02901, 2020.

Galileo Namata, Ben London, Lise Getoor, Bert Huang, and UMD EDU. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, volume 8, 2012.

Mark EJ Newman. Mixing patterns in networks. *Physical review E*, 67(2):026126, 2003.

Leto Peel. Graph-based semi-supervised learning for relational networks. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 435–443. SIAM, 2017.

Meng Qu, Yoshua Bengio, and Jian Tang. GMNN: Graph markov neural networks. In *International Conference on Machine Learning*, pp. 5241–5250, 2019.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2019.

Emanuele Rossi, Fabrizio Frasca, Ben Chamberlain, Davide Eynard, Michael Bronstein, and Federico Monti. SIGN: Scalable inception graph neural networks. *arXiv:2004.11198*, 2020.

Cosma Shalizi. *Advanced data analysis from an elementary point of view*. Cambridge University Press, 2013.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, and Yu Sun. Masked label prediction: Unified massage passing model for semi-supervised classification. *arXiv:2009.03509*, 2020.

Amanda L Traud, Peter J Mucha, and Mason A Porter. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391(16):4165–4180, 2012.

Francesco Tudisco, Austin R Benson, and Konstantin Prokopchik. Nonlinear higher-order label spreading. *arXiv:2006.04762*, 2020.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

Hongwei Wang and Jure Leskovec. Unifying graph convolutional neural networks and label propagation. *arXiv:2002.06755*, 2020.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 6861–6871. PMLR, 2019.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, C. Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48, 2016.

Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 555–564, 2017.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graph-SAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2019.

Yilin Zhang and Karl Rohe. Understanding regularized spectral clustering via graph conductance. In *Advances in Neural Information Processing Systems*, pp. 10631–10640, 2018.

Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in neural information processing systems*, pp. 321–328, 2004.

Xiaojin Zhu, Zoubin Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the International Conference on Machine Learning*, 2003.

Xiaojin Jerry Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.