Nonlinear Correct and Smooth for Semi-Supervised Learning

1st Yuanhang Shao
Department of Computer Science
Florida State University
Tallahassee, USA
shao@cs.fsu.edu

2nd Xiuwen Liu
Department of Computer Science
Florida State University
Tallahassee, USA
liux@cs.fsu.edu

2024 International Joint Conference on Neural Networks (IJCNN) | 979-8-3503-5931-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/IJCNN60899.2024.10650407

***Abstract*—Graph-based semi-supervised learning (GSSL) has been used successfully in various applications, with existing methods leveraging the graph structure and labeled samples for classification. Label Propagation (LP) and Graph Neural Networks (GNNs) both iteratively pass messages on graphs, where LP propagates and updates node labels across the graph and GNN aggregates node features from their neighboring nodes. Recently, combining LP and GNN has led to improvements in performance, yet the joint utilization of labels and features in higher-order structures of graphs, such as triangles, remains unexplored. Therefore, we introduce Nonlinear Correct and Smooth (NLCS), a combined post-processing method that incorporates non-linearity and higher-order representation into the residual propagation to address intricate node relationships effectively. Systematic evaluations show that our approach achieves remarkable average improvements of 13.2% over base prediction and 2.1% over the state-of-the-art post-processing method on six commonly used datasets. Comparisons and analyses reveal that our method enhances the prediction accuracy of nodes with complex architecture by effectively utilizing triangle relationships within graphs.**

***Index Terms*—Residual Propagation, Semi-Supervised Learning, Graph.**

I. Introduction

Graphs are indispensable in the realm of big data for representing data generated from non-Euclidean domains, as they consist of interconnected nodes carrying complex relationships and dependencies between entities. Recently, the growing variety of graph-based real-world applications, such as social networks [1], citation networks [2], and recommendation systems [3], have heightened the necessity for effective models to capture the underlying intricate relationships. Graph-based semi-supervised learning (GSSL) aims at node classification in graphs with limited labeled data, relying on the homophily [4] where neighboring nodes tend to have the same labels or similar features, a principle widely exploited in existing GSSL methods. Many recent studies in GSSL have primarily focused on graph representations using both labels and features [5]–[10]. Effectively and jointly leveraging labels, features, and higher-order representations to maximize the generalization performance of GSSL is still under exploration.

Traditional approaches in GSSL have primarily focused on utilizing homophily within labels to develop smoothing techniques, ranging from early methods like random walks [11] to more recent label propagation (LP) [12]. The standard LP algorithm spreads the known labels to adjacent neighbors through edges iteratively. Further research introduces an interpretable objective function to enhance modeling capacity by incorporating higher-order representations (such as triangles, 3-cliques, etc.) [13]. On the other hand, Graph Neural Networks (GNNs) focus on learning expressive node representations and smoothing node features [14]. Unlike fully connected networks treating inputs as an ordered list, GNNs generalize neural networks to effectively capture the complex relationships of graphs through aggregating neighbor features and smoothing to reduce noise [6], [15], [16]. Nonetheless, GNNs are sensitive to hyperparameters and lack meaningful representation for the smoothed output [15]–[18]. Recent studies have shown that a combination of label spreading with deep learning approaches can yield competitive performance [5], [7], [8]. These works emphasize the advantages of using labels as auxiliary knowledge rather than exclusively relying on features. Consequently, the development of an approach capable of learning node representation and enhancing its interpretable modeling capacity to effectively capture graph topology could further unify these two directions and strengthen existing combined methods. To date, there has been limited exploration of higher-order graph structures within the realm of combined methods involving propagation algorithms and neural networks.

To explore the aforementioned area, we propose Nonlinear Correct and Smooth (NLCS), a combined method that jointly leverages labels and features. NLCS starts by utilizing a neural network model to learn node representations as base prediction and then propagates residuals to correct labels using nonlinear and higher-order relationships. Specifically, NLCS is inspired by Correct and Smooth (C&S) [5], where our work explores how the post-processing enhances modeling ability by integrating non-linearity and higher-order representation into the residual propagation. Unlike linear propagation, the introduction of residual propagation over triangles effectively discriminates misclassified nodes to improve prediction accuracy. Through extensive analysis and experimentation, the proposed method shows improvements both locally and globally, thereby validating the effectiveness of our joint utilization of labels and features on higher-order residual propagation.

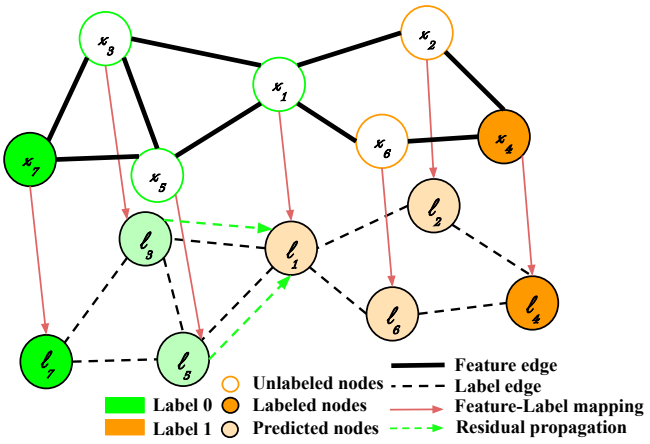


Fig. 1: Overview of NLCS. The nonlinear propagation of triangles reveals structural disparities between the left-hand and right-hand sides, a distinction not achievable through linear propagation.

II. Related Work

In recent years, significant progress in graph analysis has led to various new and improved methods for solving GSSL. Here we provide a brief review to investigate approaches in three categories: LP, GNNs, and combined methods.

Label propagation (LP) is a well-known technique for spreading known labels to unlabeled nodes based on the homophily principle. Zhou et al. [12] proposed the standard LP algorithm, which is an iterative propagation function spreading labels via the structure of graphs. Moreover, Tudisco et al. [13] proposed a nonlinear label spreading function, incorporating the higher-order structure of graphs, known as Nonlinear Higher-order Label Spreading (NHOLS). Notably, the NHOLS outperforms the GraphSAGE, a two-layer GNN that incorporates concatenation-based skip connections to alleviate over-smoothing, while NHOLS achieves superior performance by leveraging nonlinear mixing functions and spreading information over hyperedges. Prokopchik et al. [19] further extended the diffusion process of NHOLS by spreading the concatenation of features and labels on hypergraphs via a new hypergraph regularization term. These approaches indicate that the propagation technique can effectively leverage the advantages of known labels and nonlinearity.

Graph Neural Networks (GNNs) are deep neural networks designed to operate on graph-structured data utilizing the inherent structure and relationships within graphs for solving various graph-based tasks [6], [7]. GNNs generally aggregate information from neighbors to generate new node representations, which reduce noises in node features to achieve promising performance in GSSL [20], [21]. Meantime, the GNN-based models have shown indistinguishable representations of nodes across different labels caused by over-smoothing and information loss [15]–[17], [22].

Therefore, recent studies focus on understanding insights of GNNs and techniques to alleviate over-smoothing, such as skip connection, graph normalization, random dropping [15], [16], [18], [23]–[25]. Li et al. [15] highlight that the graph convolutional network (GCN) is a special form of Laplacian smoothing, which can smooth the noise of node features. However, this smoothing process can lead to less distinguishable features among different clusters. In the attempt to elucidate the effectiveness and limits of GNNs, Chen et al. [26] show that deeper GNNs are more susceptible to over-smoothing. Even though the initial and jumping connection improves the performance of deep GNNs, the normalization techniques exhibit inconsistent performance on different GNN backbones and become unstable with deeper networks; and the random dropping mitigates performance degradation but fails to improve deep GNNs consistently. Furthermore, Oono et al. [17] emphasize that a deeper network and the introduction of nonlineality can aggravate the over-smoothing issue. From these studies, there is still a trade-off between smoothing noise and making node features distinguishable, and the effectiveness of initial and jumping connections suggests the importance of preserving the original feature vectors and their correlation with labels. Intuitively, exploring alternative methods that leverage the labeled data and mitigate information loss caused by aggregation is worthwhile.

Combined methods combine GNNs with regular neural networks or LP to improve prediction accuracy, which has gained increasing interest in recent years [5], [7]–[10], [27]. Ding et al. [7] propose a framework (known as Meta-PN) that treats the predictions of an adaptive label propagator as pseudo labels and utilizes multi-layer perceptron (MLP) to predict the labels of entire graph based on these pseudo labels. On the other hand, in contrast to the learning schema of Meta-PN, Huang et al. [5] propose a different approach with their simple post-process step (C&S) based on residual propagation and smoothing to improve the base prediction of deep neural networks, including MLP and GNNs. This approach provides a new perspective for solving semi-supervised learning and opens up new avenues for exploring the combination of LP and deep neural networks. Similarly, Wang et al. [8] introduce the concept of influences and study the influences of features and labels in label propagation algorithm (LPA) and graph convolutional neural networks (GCN). They proposed a unified model to make the edges trainable based on label influence, as nodes with the same label should have strong connections. Another notable work, Jia et al. [27] propose a Markov random field model (MRF) for node attributes generation, such that the label propagation, linearized graph convolutional network, and their combination can be derived as conditional expectations under the MRF model. Han et al. [10] proposed a framework from a multi-view learning perspective consisting of a single-level optimization problem by introducing a latent variable to capture three different views of graphs: features, labels, and graph structure. Overall, label propagation proves to be a flexible method that can be applied to labels, features, residuals, and smoothing. These combined methods show great promise in utilizing labels and features jointly to solve semi-supervised learning tasks. However, none of these combined methods have explored the potential of

higher-order representation. By conducting nonlinear higher-order residual propagation (specifically via triangles) on top of base predictions, our approach NLCS achieves superior performance on GSSL.

III. Preliminary

A. Problem Definition

Given an undirected graph $G = (V, E, \omega)$, where $V = \{v_1, v_2, \ldots, v_n\}$ represents the set of n nodes, each of which can be assigned one of the labels from the set of classes $\{1, \ldots, c\}$, and $E \subseteq V \times V$ represents the set of m edges, and each edge $e \in E$ is associated with a positive weight $\omega(ij) > 0$. This set of nodes V has their corresponding feature vectors represented as $X \in \mathbb{R}^{n \times p}$. The graph-based semi-supervised classification problem aims to predict the labels of a subset of unlabeled nodes $U \subset V$ using a small, disjoint, and labeled subset $L \subset V$, based on the labels of the known subset and the features of nodes, where $U + L = V$. For this given graph G, let A be the adjacency matrix of G, such that $A_{ij} = \omega(ij), ij \in E$, and $A_{ij} = 0$ otherwise; $D_G = Diag(d_1, d_2, \ldots, d_n)$ be the diagonal degree matrix; and $S = D_G^{-\frac{1}{2}} A D_G^{-\frac{1}{2}}$ be the normalized adjacency matrix.

B. Definition of Hypergraph

Consider a 3-regular hypergraph $H = (V, \varepsilon, \tau)$ representing the weighted graph G, where V is the vertex set containing n unique vertices, $\varepsilon \in V \times V \times V$ is a set of hyperedges, where each hyperedge is a subset of V containing exactly three vertices as a triangle, and τ is the weight function of hyperedges, such that $\tau_{ijk} > 0$. For this given hypergraph H, the third-order adjacency tensor is defined as $\mathcal{A}$, where $\mathcal{A}_{ijk} = \tau_{ijk}$ and 0 otherwise. Furthermore, the diagonal matrix of the hypergraph is defined as $D_H = Diag(\delta_1, \ldots, \delta_n)$, where the vertex degree is defined as $\delta_i = \sum_{jk} \tau(ijk), ijk \in \varepsilon$.

IV. NLCS

Figure 1 illustrates the overall workflow of our proposed NLCS. In this example, vertices can be represented by their feature vectors $\boldsymbol{x} = \{x_1, \ldots, x_7\}$. Neural networks serve as base predictions to learn the representations of nodes and then map these representations to labels $\boldsymbol{l} = \{l_1, \ldots, l_7\}$. The base prediction is possible to generate wrong predictions like x_1, where its true label is 0 and prediction is 1. After the base prediction, the residual of neighboring nodes is assumed to be similar in base prediction, such that the error can propagate through a pair of edges in the triangle to distinguish two sides of the graph structure and correct the wrong prediction. Finally, NLCS applies the nonlinear smoothing over corrected predictions to get the final classification results.

A. Nonlinear Correct Stage

Our approach is a post-processing step including two stages, where the first stage aims to improve the prediction accuracy of base prediction by propagating residuals via the higher-order structure of graphs. This residual propagation assumes that the errors of neighboring nodes are likely similar to those in the base prediction results [5]. However, the neighbors are more complex in real-world graphs and not necessarily surrounded by the same labels, which can mislead the prediction, such as in Figure 2. In order to resolve the ambiguity of prediction, such as the same number of neighbors with different labels or various labeled neighbors, triangles can be a meaningful topology in the graphs from real-world applications [28]. This meaningful structure of graphs prompts that the use of higher-order residual propagation to correct the base prediction is worthwhile. First of all, NLCS uses the graph and the 3-regular graph notation defined in section III, where a 3-regular hypergraph is constructed from the existing structure of a graph, which considers the set of triangles as hyperedges of H and the cosine of the angle of triangles as the weight function of H. Then, we define an error matrix $E \in \mathbb{R}^{n \times c}$ based on the one-hot-encoding matrix $Y \in \mathbb{R}^{n \times c}$ regarding the node labels, where c is the number of labels and $Y_{ij} = 1$ if node $i \in L$ for label j, and 0 otherwise, which is split into two parts:

$$E_L = X_L - Y_L, E_U = 0, \tag{1}$$

where X_L is the base prediction of labeled nodes, and the error is the residual on labeled and 0 on unlabeled. Then, we propagate errors at node i to neighboring nodes of i which should increase the similar error to surrounding nodes. We utilize the nonlinear label spreading technique of Tudisco et al. [13] by using the nonlinear mixing functions $\sigma : \mathbb{R}^2 \to \mathbb{R}$ to define the tensor mapping $F : \mathbb{R}^{n \times c} \to \mathbb{R}^{n \times c}$ regarding the error matrix entrywise as follows:

$$F(E)_i = \sum_{jk} \mathcal{A}_{ijk} \sigma(E_j, E_k). \tag{2}$$

This tensor mapping aggregates errors from neighbors of triangles via a nonlinear mixing function, such as triangles which consider errors from two adjacency nodes of a triangle and its cosine of the angle. Then, the nonlinear mapping for the error matrix is denoted by:

$$\mathcal{S}(E) = D_H^{-\frac{1}{2}} F(D_H^{-\frac{1}{2}} E). \tag{3}$$

At last, the residual propagation is iterated by:

$$E^{(t+1)} = \alpha \mathcal{S}(E^{(t)}) + \beta S E^{(t)} + (1 - \alpha - \beta) Y, \tag{4}$$

where the initial vector $E_{(0)} = E$. This iteration function propagates the error in third-order adjacency representation or higher, in which the higher-order propagation is nonlinear. The error propagation is provable under a Gaussian assumption in regression problems [29], and the errors in equation 4 need normalization to adjust the scale of residual. We adopt the Autoscale [5], in which the scale of the error is approximated with the average error of the labeled nodes. The L1-norm for labeled nodes is denoted as:

$$\lambda = \frac{1}{|L|} \sum_{j \in L} ||e_j||_1, \tag{5}$$

where $e_j \in \mathbb{R}^c$ is the jth row of E. Eventually, the base prediction of unlabeled nodes is corrected by the error that

is propagated through the higher-order nonlinear function as follows:

$$X'_i = X_i + \lambda \frac{\widehat{E}_i}{||\widehat{E}_i||_1}, i \in U, \tag{6}$$

where $\widehat{E}_i$ is the approximated error of ith unlabeled node after the error propagation iteration in equation 4.

B. Nonlinear Smooth Stage

After the correction using propagated residual, our approach conducts smoothing based on corrected labels. Considering the diversity of graphs, the residual correction handles heterophily neighbors to provide good estimations. The corrected predictions of unknown labels are considered relatively accurate compared to base predictions. Therefore, smoothing can refine and improve the overall prediction accuracy by applying additional label propagation over the entire graph. This step utilizes the best predicted labels $G \in \mathbb{R}^{n \times c}$:

$$G_L = Y_L, G_U = X'_U, \tag{7}$$

where the labeled nodes use the true label of training and the unlabeled nodes use the corrected prediction in equation 6. Subsequently, the final prediction $\hat{Y}$ is approximated iteratively using the nonlinear higher-order label spreading function $G^{(t+1)} = (\alpha \mathcal{S}(G^{(t)}) + \beta S G^{(t)} + (1-\alpha-\beta)Y)/\varphi(G^{(t)})$, where $G^{(0)} = G$ and φ denotes the normalization term [13]. Finally, for each node $i \in U$, the prediction of the label is assigned as the one with the highest probability among the classes denoted as $argmax_{j \in \{1,2,\ldots,C\}} \hat{Y}_{ij}$.

Application	Dataset	$\lvert\mathcal{N}\rvert$	$\lvert E\rvert$	$\lvert L\rvert$
Social	Rice31	3,560	317,828	9
Social	Caltech36	590	25,644	8
Citation	arxiv	169,343	2,315,598	40
Citation	Cora	2,708	10,556	7
Citation	CiteSeer	3,327	9,104	6
Citation	PubMed	19,717	88,648	3

TABLE I: Dataset and graph size.

V. Experiment Evaluation

In this section, we conduct empirical evaluations to assess the performance of the NLCS. Our analysis aims to investigate the following questions:

Question 1 (Q1): Can the utilization of the nonlinear higher-order structure of graphs improve performance in real-world graphs?

Question 2 (Q2): How does the higher-order representation affect the prediction results in each stage of NLCS?

Question 3 (Q3): How do the base prediction and post-processing steps influence each other and end up with such improvement?

In order to answer these questions, NLCS is compared against C&S over three base prediction models and six datasets. We performed an in-depth analysis of the NLCS by analyzing the distribution of output values from both network models and each post-processing stage. Additionally, we investigate the impact of post-process steps on base prediction via the training process. By conducting thorough experiments, we aim to gain insights into the performance and interpretability of the proposed method.

A. Experiment Setup

Baselines: We compare NLCS to several baselines including propagation algorithms and network models: standard label propagation (LP) [12], nonlinear higher-order label spreading (NHOLS) [13], plain linear (PL), multi-layer perceptron (MLP), and graph attention network (GAT) [30]. For each of the baseline methods, we train on 10 random initialized network models and apply both our post-processing step and C&S to analyze their performance via six commonly used datasets. We provide the source code link[1] for our experiments.

Datasets: We evaluate the performance of each baseline using the datasets listed in Table I, where Rice31 and Caltech36 are provided in source code[1] and other datasets are public by Yang et al. [31] and Hu et al. [32]. For each dataset, we split it into train, validation, and test subsets based on the ratio of known labels (k). The split is performed by randomly selecting an equal number of samples from each class, with ratio k, $\frac{(1-k)}{2}$, and $\frac{(1-k)}{2}$ respectively. The same split is applied consistently across different baselines for a fair comparison. Considering the size of Caltech36, the percentages of known labels are set to 10% and 20% in Table II, while the other dataset remains 5% and 10%. It should be noted that the social network datasets do not include node features, whereas the citation network datasets are the opposite.

Parameters: We systematically explore the constant parameters α and β in Eq. 4 by varying them from range 0 to 1, with increments of 0.1. The provided source code includes the optimal parameter settings that we have discovered through extensive experimentation. Additionally, for the propagation function, we employ a fixed constant number of iterations ($t = 50$), which has proven to be sufficient for convergence. Regarding the training of neural network models, all experiments run for 1000 epochs with learning rates ranging from 0.01 to 0.001, depending on the specific dataset. The multi-layer models (MLP and GAT) consist of 256 hidden neurons and a dropout rate of 0.5.

B. Performance Evaluation (Q1, Q2)

We compare the performance of base predictions after applying our post-processing step with C&S and other neural network models. As shown in Table II, NLCS improves the average accuracy by 19.77%, 11.14%, and 8.69% across all datasets when compared to PL, MLP, and GAT base predictions, respectively. On the other hand, the C&S improves the average accuracy by 16.96%, 9.19%, and 7.82% across all datasets with PL, MLP, and GAT base prediction. The plain linear network model exhibits low accuracy on all datasets when the node features are unavailable as expected. Meanwhile, the MLP and GAT effectively learn the node feature representations for classification.

[1]Code Repository: https://gitlab.com/Shao1206/nlcs

Application	Rice31		Caltech36		Cora	
	5%	10%	10%	20%	5%	10%
LP	80.80	87.51	70.38	78.03	69.68	74.49
NHOLS	86.95	89.93	82.41	85.63	65.99	72.14
PL	58.19 ± 0.10	71.83 ± 0.32	41.08 ± 0.63	49.80 ± 0.51	46.94 ± 0.06	54.94 ± 0.04
PL+C&S	82.31 ± 0.10	88.51 ± 0.10	76.58 ± 1.56	79.83 ± 0.20	60.11 ± 0.11	79.02 ± 0.20
PL+NLCS	**87.44 ± 0.05**	**90.77 ± 0.06**	**81.88 ± 0.47**	**84.62 ± 0.15**	**77.22 ± 0.08**	**80.71 ± 0.05**
MLP	69.33 ± 0.66	79.84 ± 0.61	46.93 ± 1.88	57.27 ± 1.81	58.74 ± 0.86	63.80 ± 0.81
MLP+C&S	72.72 ± 0.97	81.69 ± 0.67	72.49 ± 2.14	75.58 ± 1.23	75.32 ± 1.08	**80.48 ± 0.74**
MLP+NLCS	**78.35 ± 0.90**	**85.16 ± 0.46**	**80.58 ± 0.82**	**83.92 ± 0.73**	**75.76 ± 0.51**	80.26 ± 0.37
GAT	73.20 ± 0.27	81.15 ± 0.62	30.95 ± 1.39	47.55 ± 1.63	78.40 ± 0.32	80.93 ± 0.66
GAT+C&S	78.32 ± 0.70	84.86 ± 0.60	79.20 ± 1.11	80.51 ± 0.77	79.35 ± 0.46	**81.51 ± 0.60**
GAT+NLCS	**82.65 ± 0.45**	**87.09 ± 0.30**	**80.60 ± 0.41**	**84.70 ± 0.48**	**79.51 ± 0.33**	81.26 ± 0.44
Application	CiteSeer		Arxiv		PubMed	
	5%	10%	5%	10%	5%	10%
LP	49.01	53.43	65.48	67.62	79.12	79.94
NHOLS	46.69	51.87	64.58	66.67	76.30	79.92
PL	52.28 ± 0.08	58.50 ± 0.19	48.53 ± 0.06	50.15 ± 0.02	76.91 ± 0.03	79.52 ± 0.03
PL+C&S	61.70 ± 0.11	64.80 ± 0.12	**67.24 ± 0.04**	**68.55 ± 0.01**	**81.75 ± 0.04**	**81.83 ± 0.01**
PL+NLCS	**62.81 ± 0.09**	**65.64 ± 0.13**	65.72 ± 0.05	67.99 ± 0.01	80.76 ± 0.07	80.32 ± 0.06
MLP	50.00 ± 1.38	63.28 ± 0.83	68.33 ± 0.10	70.16 ± 0.05	79.16 ± 0.25	81.88 ± 0.33
MLP+C&S	59.72 ± 1.14	**69.20 ± 0.94**	**70.80 ± 0.08**	**72.35 ± 0.07**	**83.43 ± 0.25**	**85.22 ± 0.22**
MLP+NLCS	**62.62 ± 0.57**	68.08 ± 0.89	69.39 ± 0.08	71.29 ± 0.13	82.67 ± 0.32	84.26 ± 0.15
GAT	64.26 ± 0.77	67.19 ± 0.49	70.13 ± 0.17	71.87 ± 0.08	84.20 ± 0.26	85.60 ± 0.25
GAT+C&S	**65.34 ± 0.63**	**67.48 ± 0.57**	**71.24 ± 0.13**	**72.69 ± 0.05**	84.34 ± 0.21	85.13 ± 0.12
GAT+NLCS	64.94 ± 0.53	66.97 ± 0.58	70.44 ± 0.12	72.17 ± 0.04	**84.43 ± 0.24**	**85.62 ± 0.24**

TABLE II: The prediction accuracy on six real-world datasets.

Application	Rice31		Caltech36		Cora	
	5%	10%	10%	20%	5%	10%
PL	58.19 ± 0.10	71.83 ± 0.32	41.08 ± 0.63	49.80 ± 0.51	46.94 ± 0.06	54.94 ± 0.04
+LC	73.36 ± 0.10	84.34 ± 0.24	72.64 ± 2.28	77.41 ± 1.50	55.06 ± 0.11	68.80 ± 0.49
+C&S	82.31 ± 0.10	88.51 ± 0.10	76.58 ± 1.56	79.83 ± 0.20	60.11 ± 0.11	79.02 ± 0.20
+NLC	80.57 ± 0.27	88.04 ± 0.24	71.47 ± 1.73	77.71 ± 1.78	54.96 ± 0.14	68.43 ± 0.55
+NLCS	**87.44 ± 0.05**	**90.77 ± 0.06**	**78.08 ± 0.78**	**81.88 ± 0.47**	**77.22 ± 0.08**	**80.71 ± 0.05**
Application	CiteSeer		Arxiv		PubMed	
	5%	10%	5%	10%	5%	10%
PL	52.28 ± 0.08	58.50 ± 0.19	48.53 ± 0.06	50.15 ± 0.02	76.91 ± 0.03	79.52 ± 0.03
+LC	58.62 ± 0.08	62.76 ± 0.04	61.43 ± 0.20	63.88 ± 0.02	81.88 ± 0.09	82.01 ± 0.04
+C&S	61.70 ± 0.11	64.80 ± 0.12	**67.24 ± 0.04**	**68.55 ± 0.01**	**81.75 ± 0.04**	**81.83 ± 0.01**
+NLC	59.83 ± 0.12	63.91 ± 0.08	57.27 ± 0.19	61.09 ± 0.02	78.90 ± 0.08	79.21 ± 0.05
+NLCS	**62.81 ± 0.09**	**65.64 ± 0.13**	65.72 ± 0.05	67.99 ± 0.01	80.76 ± 0.07	80.32 ± 0.06

TABLE III: The performance on each post-processing step using linear and nonlinear, where LC means linear correction and NLC means nonlinear correction.

Notably, the performance of two social network datasets is further improved by 5.27% when applying our nonlinear higher-order error spreading compared to the C&S across all three base predictions. This improvement is particularly significant because social network datasets lack node feature information, limiting the performance of MLP and GAT models. In this case, the combination of base prediction with nonlinear post-processing surpasses the performance of sophisticated neural network models.

In the case of citation networks, NLCS achieves an average improvement of 5.19% on Cora and CiteSeer compared to C&S with PL. The performance of NLCS is competitive against C&S using MLP and GAT base predictions. However, when comparing our results to citation network baselines, it is noteworthy that NLCS is marginally below C&S by an average of 0.79% in Arxiv and PubMed datasets. These results are consistent with research showing that both NLCS and C&S exhibit the same performance patterns across different real-world datasets. A possible explanation is that there are relatively sparse graphs with a lower node-to-edge ratio where NLCS has less higher-order structure to utilize. Overall, NLCS improves the performance by 7.3% on average across all three base prediction models in all four citation datasets.

From these demonstrated results, the post-processing step yields greater improvements when the node features are not directly associated with labels, and the network models struggle to learn the representation of nodes. This observation aligns with the notation of feature and label influence that has been discussed in [8]. Moreover, NLCS shows its effectiveness by yielding lower standard deviation (STD) in 72.2% of cases and higher accuracy in 58.3% of cases compared to C&S. This indicates that NLCS provides more consistent and reliable

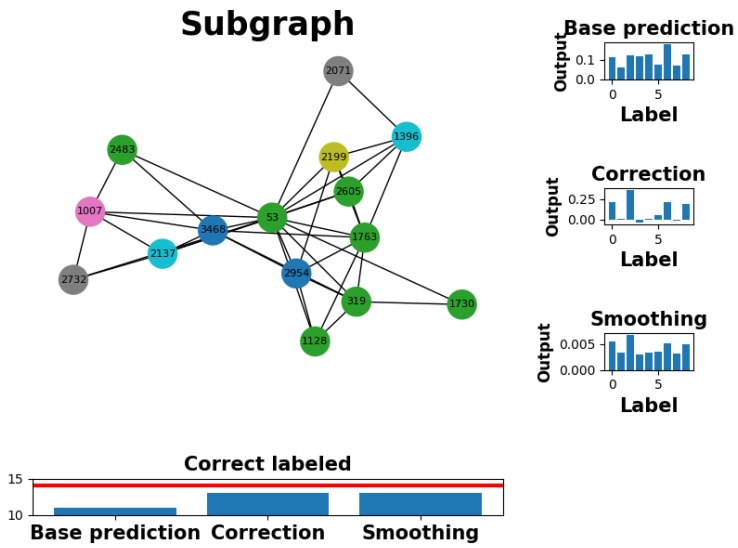
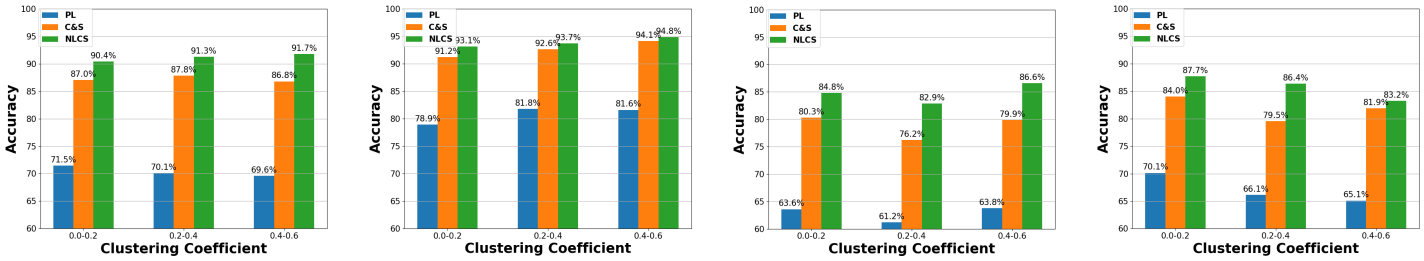


Fig. 2: The subgraph visualization showcases the NLCS results of center node 53 and its one-hop neighbors from the Rice31 dataset with 10% known labels.



(a) Rice31-5% (b) Rice31-10% (c) Cora-5% (d) Cora-10%

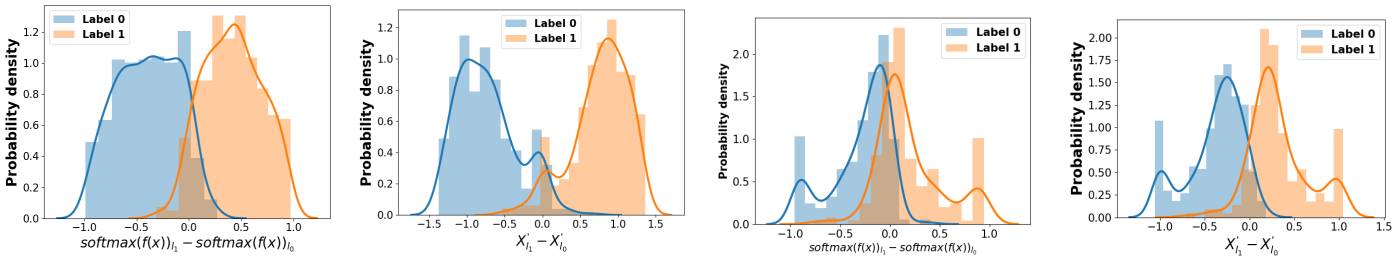
Fig. 3: The prediction accuracy at different ranges of clustering coefficient.

results with improved prediction accuracy utilizing higher-order representation.

C. Distribution Analysis (Q3)

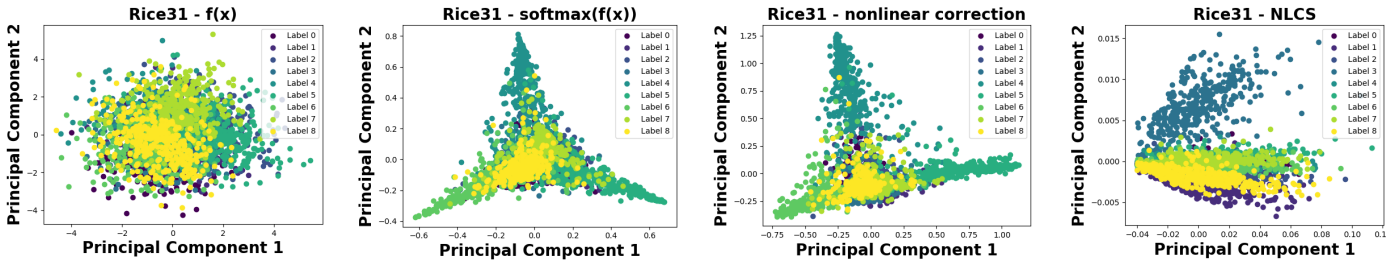
To gain insights into the effectiveness of the post-processing step, we conducted an output value distribution analysis at individual nodes and their neighbors. As shown in Figure 2, we select an individual node (Node 53) from the Rice31 dataset and illustrate the output value distribution of the center node regarding different classes at each step on the right-hand side (base prediction, nonlinear correction, and nonlinear smoothing, respectively). Additionally, we provide the number of correct predictions for its adjacency nodes at the bottom, where the red line indicates the total number of neighbors. Based on the observation of this subgraph, it is obvious that the neighbors of this individual node belong to multiple classes, which makes accurate predictions challenging. Remarkably, for this individual node, even though both the node itself and its neighbors are unlabeled, our method successfully predicts 13 out of 14 neighbors in this subgraph. This success is due to the prevalent triangular structure observed in the subgraph, which our method utilizes effectively by leveraging higher-order representation to improve prediction accuracy. Moreover, we analyze the prediction accuracy corresponding to the ranges of clustering coefficient from 0 to 0.6 in Figure 3. This range corresponds to the majority of nodes in the dataset. Notably, NLCS consistently outperformed both the base prediction and C&S methods across different coefficients. Overall, by incorporating higher-order representation in the correction and smoothing approaches, our methodology shows modeling flexibility in complex relationships while improving performance consistently.

We further analyzed the output value distribution from an overall perspective at each stage of our method: base prediction, correction, and smoothing. In Figure 4, we observe the classification change by selecting the first two classes of nodes and visualizing the difference between the first two output values. Figures 4a and 4c illustrate that the base prediction is capable of distinguishing between classes based on the feature vectors. However, the misclassification happens when the difference value is greater than 0 in Label 0 and smaller than 0 in Label 1. Figure 4b - 4d demonstrates that the overlapping area decreases after correction and smoothing, indicating a reduction in incorrect classifications and an improvement in accuracy. This trend is more significant in the analysis of the



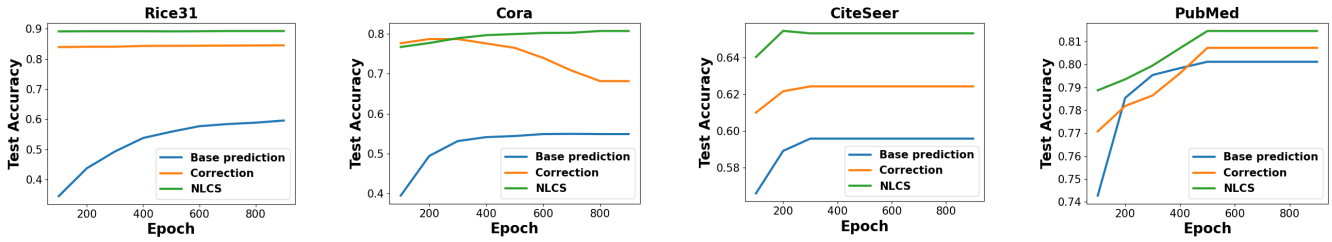
(a) Rice31 (b) Rice31 (c) Cora (d) Cora

Fig. 4: The distribution of the difference value between the first two output values regarding base prediction and correction for the Rice31 and Cora datasets.



(a) (b) (c) (d)

Fig. 5: The PCA visualization on each step's output of Rice31 with 5% known labels.



(a) (b) (c) (d)

Fig. 6: The accuracy during the training to the base prediction model, nonlinear correction, and NLCS.

Cora dataset, as shown in Figure 4c to 4d.

To intuitively understand how our method clusters nodes in a spatial domain, we visualize the multidimensional output vectors of prediction using principal component analysis (PCA). The results of Figure 5 demonstrate the 2-dimensional plots of the output value from linear layer, PL model, correction, and smoothing. The results show that the neural network learns the representation of nodes and provides a baseline of accuracy. Subsequently, the nonlinear correction further brings the cluster to a more confident level, and the nonlinear smoothing compresses the information into a lower dimension.

D. Accuracy Timeline and Complexity

We analyzed the accuracy during the training process of the base prediction and after applying NLCS post-processing steps to assess performance stability. Figure 6 shows the testing accuracy during the training and applies NLCS on every 100 epochs. The plots show that our post-processing steps consistently improve the performance over time. Although the specific benefits gained from these two steps may vary depending on the dataset, the overall performance trend remains the same. This suggests that NLCS is robust and stable, even when the base prediction is not well-trained.

Considering the hypergraph definition in section III-B, our method entails an increased computational complexity, which is attributed to both the initial construction of hypergraphs and the subsequent error propagation steps. The construction of the hypergraph is executed once and carries a time complexity of $O(\frac{Tr(D_G)^2}{|V|})$, where $Tr(D_G) = \sum_{i=1}^{n}(D_G)_{ii}$ is the sum degree of nodes. The higher-order error propagation has a

linear time complexity of $O(|\varepsilon|)$ with respect to the number of hyperedges ε which directly correlates with the number of triangles. These complexities underscore the impact of the graph structure, specifically, the higher-order structure connections, on the computational demands of our method.

VI. Conclusion

We have demonstrated the significant potential of incorporating nonlinearity and higher-order structures in post-processing steps for GSSL. By effectively utilizing both labels and features in a higher-order manner, NLCS exhibits more effective modeling capabilities compared to C&S and advances the exploration of joint utilization in this context. Our results have shown substantial advantages of incorporating higher-order structures in graph-based learning. The proposed NLCS can be flexibly employed as a post-processing step after various neural network models, showcasing a broader applicability of the higher-order concept in the graph research area. Our results indicate that both nonlinear and linear methods show varying degrees of effectiveness across different datasets. This variability provides a future research direction: a unified framework capable of discerning the unique characteristics of local graph structures and customizing solutions to optimize performance. As we continue to explore this direction, we anticipate further advancements in the field, preparing the way for more sophisticated and efficient graph learning methods.

References

[1] C. T. Butts, "Social network analysis: A methodological introduction," *Asian Journal of Social Psychology*, vol. 11, no. 1, pp. 13–41, 2008.

[2] S. Dawson, D. Gašević, G. Siemens, and S. Joksimovic, "Current state and future trends: A citation network analysis of the learning analytics field," in *Proceedings of the fourth international conference on learning analytics and knowledge*, 2014, pp. 231–240.

[3] D. A. Adeniyi, Z. Wei, and Y. Yongquan, "Automated web usage data mining and recommendation system using k-nearest neighbor (knn) classification method," *Applied Computing and Informatics*, vol. 12, no. 1, pp. 90–108, 2016.

[4] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," *Advances in neural information processing systems*, vol. 33, pp. 7793–7804, 2020.

[5] Q. Huang, H. He, A. Singh, S.-N. Lim, and A. R. Benson, "Combining label propagation and simple models out-performs graph neural networks," *arXiv preprint arXiv:2010.13993*, 2020.

[6] V. Verma, M. Qu, K. Kawaguchi, A. Lamb, Y. Bengio, J. Kannala, and J. Tang, "Graphmix: Improved training of gnns for semi-supervised learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 11, 2021, pp. 10 024–10 032.

[7] K. Ding, J. Wang, J. Caverlee, and H. Liu, "Meta propagation networks for graph few-shot semi-supervised learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 6, 2022, pp. 6524–6531.

[8] H. Wang and J. Leskovec, "Combining graph convolutional neural networks and label propagation," *ACM Transactions on Information Systems (TOIS)*, vol. 40, no. 4, pp. 1–27, 2021.

[9] Y. Wang, J. Jin, W. Zhang, Y. Yang, J. Chen, Q. Gan, Y. Yu, Z. Zhang, Z. Huang, and D. Wipf, "Why propagate alone? parallel use of labels and features on graphs," *arXiv preprint arXiv:2110.07190*, 2021.

[10] H. Han, X. Liu, H. Mao, M. Torkamani, F. Shi, V. Lee, and J. Tang, "Alternately optimized graph neural networks," in *International Conference on Machine Learning*. PMLR, 2023, pp. 12 411–12 429.

[11] M. Szummer and T. Jaakkola, "Partially labeled classification with markov random walks," *Advances in neural information processing systems*, vol. 14, 2001.

[12] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," *Advances in neural information processing systems*, vol. 16, 2003.

[13] F. Tudisco, A. R. Benson, and K. Prokopchik, "Nonlinear higher-order label spreading," in *Proceedings of the Web Conference 2021*, 2021, pp. 2402–2413.

[14] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[15] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Thirty-Second AAAI conference on artificial intelligence*, 2018.

[16] W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang, "Graph random neural networks for semi-supervised learning on graphs," *Advances in neural information processing systems*, vol. 33, pp. 22 092–22 103, 2020.

[17] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," *arXiv preprint arXiv:1905.10947*, 2019.

[18] G. Li, M. Muller, A. Thabet, and B. Ghanem, "Deepgcns: Can gcns go as deep as cnns?" in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9267–9276.

[19] K. Prokopchik, A. R. Benson, and F. Tudisco, "Nonlinear feature diffusion on hypergraphs," in *International Conference on Machine Learning*. PMLR, 2022, pp. 17 945–17 958.

[20] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[21] Y. Chong, Y. Ding, Q. Yan, and S. Pan, "Graph-based semi-supervised learning: A review," *Neurocomputing*, vol. 408, pp. 216–230, 2020.

[22] K. K. Thekumparampil, C. Wang, S. Oh, and L.-J. Li, "Attention-based graph neural network for semi-supervised learning," *arXiv preprint arXiv:1803.03735*, 2018.

[23] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *International conference on machine learning*. PMLR, 2020, pp. 1725–1735.

[24] K. Zhou, Y. Dong, K. Wang, W. S. Lee, B. Hooi, H. Xu, and J. Feng, "Understanding and resolving performance degradation in deep graph convolutional networks," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 2728–2737.

[25] W. Huang, Y. Rong, T. Xu, F. Sun, and J. Huang, "Tackling over-smoothing for general graph convolutional networks," *arXiv preprint arXiv:2008.09864*, 2020.

[26] T. Chen, K. Zhou, K. Duan, W. Zheng, P. Wang, X. Hu, and Z. Wang, "Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[27] J. Jia and A. R. Benson, "A unifying generative model for graph learning algorithms: Label propagation, graph convolutions, and combinations," *SIAM Journal on Mathematics of Data Science*, vol. 4, no. 1, pp. 100–125, 2022.

[28] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015. [Online]. Available: https://networkrepository.com

[29] J. Jia and A. R. Benson, "Residual correlation in graph neural network regression," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 588–598.

[30] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[31] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *International conference on machine learning*. PMLR, 2016, pp. 40–48.

[32] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *Advances in neural information processing systems*, vol. 33, pp. 22 118–22 133, 2020.