Forming an Electoral College for a Graph: a Heuristic Semi-supervised Learning Framework

Chen Li  
Beihang University  
lichen@act.buaa.edu.cn

Xutan Peng  
The University of Sheffield  
x.peng@shef.ac.uk

Hao Peng  
Beihang University  
penghao@buaa.edu.cn

Jianxin Li  
Beihang University  
lijx@act.buaa.edu.cn

Lihong Wang  
CNCERT  
wlh@isc.org.cn

Philip S. Yu  
UIC  
psyu@uic.edu

Abstract

Recently, graph-based algorithms have drawn much attention because of their impressive success in semi-supervised scenarios. For better model performance, previous studies learn to transform the topology of the input graph. However, these works only focus on optimizing the original nodes and edges, leaving the direction of augmenting existing data unexplored. In this paper, by simulating the generation process of graph signals, we propose a novel heuristic pre-processing technique, namely ELectoral COllege (ELCO), which automatically expands new nodes and edges to refine the label similarity within a dense subgraph. Substantially enlarging the original training set with high-quality generated labeled data, our framework can effectively benefit downstream models. To justify the generality and practicality of ELCO, we couple it with the popular Graph Convolution Network and Graph Attention Network to extensively perform semi-supervised learning evaluations on three standard datasets. In all setups tested, our method boosts the average score of base models by a large margin of 4 points, as well as consistently outperforms the state-of-the-art. Please find our code at https://github.com/RingBDStack/ELCO.

1 Introduction

Numerous real-world data can be represented as graphs, e.g., social networks [18], citation networks [29], knowledge graphs [22], and protein-interaction networks [6]. In many cases, large-scale annotated data is expensive to obtain. The so-called graph-based Semi-Supervised Learning (SSL), which holds promise to bootstrap applications even with limited supervision, has therefore attracted increasing research interest.

Earlier works develop the classical regularization methods, which achieve SSL by smoothing feature representations or model predictions over local neighborhoods using explicit regularization schemes [2, 13, 15, 32]. Although this direction has been well studied, a later thread of algorithms, namely graph convolution network methods, has demonstrated state-of-the-art performance and drawn much attention [3, 11, 26]. By utilizing various aggregation strategies, these models selectively fuse the local features of the graph into the hidden representations of its target nodes. To further perform downstream tasks, the hidden layers are coupled with specific task layers [11, 26]. One common characteristic of these two strands of models is that, they both adopt the presence of smoothness within the graph structure as a basic assumption.

Preprint. Under review.
Recently, to better exploit annotated resources, some studies propose to modify the topology of the input graph. For instance, DropEdge [20] prevents excessive smoothing by simplifying edges (i.e., randomly dropping a certain number of edges from the given graph); Abu-El-Haija et al. [1] adjust the local distribution of nodes by repeatedly mixing neighbourhoods at various scales; Yang et al. [28] restructure the graph based on modularity, thus strengthening the intra-community connections but reducing the inter-community ones. However, to the best of our knowledge, all such methods are limited within handling the existing graph topology.

In this paper, we explore a novel research direction for the first time, which aims to expand the original graph by generating new nodes and edges. More concretely, our ELECToral COllege (ELCO) framework, which is inspired by the widely-known Electoral College system of the United States, first identifies dense subgraphs (constituency divisions) through overlapping clustering algorithms. Consequently, for each subgraph, by jointly considering node attributes and edge links, it generates an elector node (elector) with attribute and label learned via the originally labeled voter nodes (voters). Lastly, ELCO connects elector nodes with their corresponding voter nodes, yielding an updated graph. As pointed out by Yang et al. [28], for a given graph, higher overall label similarity within the same dense subgraphs indicates better performance in subsequent tasks. From the generative perspective of graph signals (see §2), theoretically and empirically we justify that our newly-generated graph is superior to the original one in terms of the aforementioned similarity. In addition, we find that our simple preprocessing technique also strengthens the class separability of node attributes (we further empirically confirm this claim in §4.3). For instance, Fig. 1 illustrates a toy sample of the augmented graph based on the Cora dataset [29], from which we observe that the attribute class separability of \( \{N_1, N_2\} \) is much stronger than that of any voter node pair (i.e., \( \{n_i, n_j\} \subseteq G_1 \cup G_2 \)). Therefore, by producing a high-quality augmented training set, ELCO generally renders the performance of subsequent SSL models stronger.

To validate the practical usefulness of ELCO, we perform extensive evaluations on the SSL benchmark with three standard datasets. Coupled with two popular models (i.e., Graph Convolutional Network (GCN) and Graph Attention Network (GAT)), our method significantly improves the performance of the base algorithms and constantly outperforms the state-of-the-art. Moreover, we conduct comprehensive experiments and statistical analyses to understand the mechanism of our method.

In summary, the contribution of this work is four-fold.

- We propose the first algorithm which learns to expand the original training set with new nodes and edges.
- By simulating the generation process of graph signals, theoretically and empirically the new data augmented by our method is shown to exhibit high quality.
- This pre-processing technique is fully agnostic to the input typologies and the subsequent systems, thus can be coupled with various graph-based SSL models.
- In the extensive evaluations of SSL on three standard datasets, our method consistently enhances two popular base algorithms and sets new state-of-the-art performance.

2 Background

For notation purposes, we first formalize the data structure of graph. Next, we introduce a generative viewpoint for graph signals, which is crucial for obtaining further theoretical insights regarding the proposed algorithm.
Data structure of graph. Formally, a attributed graph $G$ (either directed or undirected) can be denoted as $\{V, E, X, Y\}$, where $V = \{v_i\}$ is a set of $|V|$ vertices in $G$, $E \in \mathbb{R}^{n \times n}$ is the adjacency relationships between vertices representing the topology of $G$ (i.e., the edge set of $V \times V$), $X \in \mathbb{R}^{n \times d}$ is the $d$-dimensional attribute matrix, and $Y \in \mathbb{R}^{n \times k}$ records the outcome/prediction vectors (with $k$ classes for each vertex label). $\forall e_{(v_i, v_j)} \neq 0, e_{(v_i, v_j)} \in E$ denotes there is an edge between $v_i$ and $v_j$, otherwise $e_{(v_i, v_j)} = 0$. In particular, when $G$ is a directed graph or $E$ contains edge weights, $e_{(v_i, v_j)} \neq e_{(v_j, v_i)}$. As a matrix with vertex/node attributes, the $i$th row in $X$ corresponds to the specific attribute of $v_i$, which can be regarded as a feature vector with signals from $d$ different channels. Given $y_i$ is a discrete one-hot label vector in matrix $Y$, it also corresponds to vertex $v_i$, and the $i$th row of $Y$.

A generative view of graph. Early works often treat the graph data as a fixed observation \cite{24,27}. However, recent studies show that this perspective has limitations, e.g., in Fig. 1 we see that the observation of $P(X|Y)$ may contain non-negligible noise. Similarly, in practice node attributes and edge links may not correspond to the likelihood of label similarity, i.e., the observation of $P(E|X,Y)$ can be far from the golden distribution, especially in the SSL setting (with only a small number of labeled nodes available) \cite{10,24}. For a better real-world approximation, researchers start to view graph data as signals generated from the ground-truth node attributes $X$ and labels $Y$, which can be described by the following factorization of the joint distribution \cite{14}:

$$P(E, X, Y) = P(E|X, Y)P(Y|X)P(X), \quad (1)$$

where the given graph $G$ is treated as a observation of $P(E, X, Y)$, and $P(E|X, Y)$ is the conditional probability of $E$ given $X$ and $Y$. Because $X$ and $Y$ are not independent, based on the conditional probability formula we have

$$P(E, X, Y) = P(E|X, Y)P(X|Y)P(Y), \quad (2)$$

where $X$ can be regarded as the generated data using $Y$.

3 Methodology

By implying the widely-adopted assumption of existing graph-based SSL models, i.e., labels exhibit smoothness along the graph edges \cite{24,27} (cf. §1), Yang et al. \cite{28} propose a criterion to assess training samples, which is highly correlated to the subsequent modeling performance:

Criterion $\breve{C}$: The more nodes in the same dense subgraphs are likely to share similar labels, the better the performance of downstream algorithms will achieve.

This criterion, which is intuitively obvious given the observed presence of graph node communities, has been empirically validated by the experiments of Yang et al. \cite{28}. Therefore, the ultimate objective of our algorithm boils to: compared with the original graph $G$, the augmented data should satisfies Criterion $\breve{C}$ equally well, or even better.

As illustrated in Fig. 2 to achieve this goal, in the first step (§3.1) we learn to partition the original graph into different dense subgraphs (i.e., clusters). Next, for each cluster, we automatically generate an elector node (§3.2), whose attributes can be regarded as the multiple sampling results on attributes of existing voter nodes. Considering the fact that multiple sampling of distribution can stabilize its
posterior probability, compared with voter nodes, elector nodes naturally have better class separability in terms of attributes (which is empirically proven by answering Question 1 in § 4.3). Consequently, while labels of some elector nodes that can be directly inherited from the voter nodes, the labels of remaining elector nodes can also be roughly determined even with a very simple classifier. Lastly (§ 3.3), since most edges between elector nodes and their corresponding voter nodes can maintain label consistency, the updated graph \( G' \) is deemed to have high quality (cf. Criterion 6) and has much larger volume than the original \( G \), thus effectively benefiting subsequent graph-based algorithms. By answering Question 2 in § 4.3, we further present valid evidence to show that in terms of the satisfaction of Criterion 6, \( G' \) is even better than \( G \).

One high-level view of the entire ELCO pipeline is that, it is actually a continuation of the original generation process of the given graph \( G \) (cf. § 2). As discussed in § 2, \( G \) is generated by a random process with \( X \) and \( Y \) as initial signals. If \( X' \) and \( Y' \) are generated from \( X \) and \( Y \) based on a specific strategy, then \( P(X'|Y')P(Y') = P(X|Y)P(Y) \). Meanwhile, the new edges in \( E' \) can be viewed as an extension or self-loop of \( E \). Therefore, in essence, the \( G' \) generated by ELCO also uses \( X \) and \( Y \) as the initial signals, thus can be regarded as the second generation of \( G \).

We detail the pipeline of the proposed ELCO as follows.

### 3.1 Constituency Division: Substructure-based Overlapping Clustering

In real-world scenarios, it is quite common that a node belongs to multiple communities (dense subgraphs), e.g., an author publishes a highly impacted paper on machine learning theory, which may get cited by different communities such as computer vision and natural language processing. Therefore, different from Yang et al. [28] who utilize non-overlapping partitioning approaches, in this paper we identify subgraphs thorough overlapping clustering, which can be classified as a community detection technique. More concretely, we adopt a robust and noise-tolerate substructure-based tool, namely Ego-Splitting [5]. The workflow of Ego-Splitting is two-phased (please refer to Appendix A for more detailed descriptions). In its first phase, Ego-Splitting learns to cluster nodes within local regions. To handle nodes belonging to multiple neighborhoods, it will create personas for each cluster. In the second phase, Ego-Splitting performs a standard global clustering and re-associates the personas whose sources are same. We denote the resulting set of dense subgraphs as \( S' \).

### 3.2 Electoral College: High-Level Information Diffusion

From the aforementioned generative perspective of graph, original voter nodes in a cluster \( C_m \in S' \) become \( |C_m| \) samples from the golden distribution of attributes. Similarly, the attributes of the corresponding elector node \( x^e_i \in X^e \) can be generated through multiple samplings, i.e., aggregating the attributes of voter nodes as

\[
x^e_i = \frac{\sum_{j \in C_m} x_{v_j}}{|C_m|}, x_{v_j} \in X
\]

where \( X^e \) denotes the attribute matrix of all elector nodes, and \( Y^e \) is for the label matrix, likewise. To determine the values of \( Y^e \), we proceed our discussions case by case.

"Winner takes all". For each elector node, if its voter nodes are from the original training set, i.e., have annotated labels, then it can straightforward inherit the dominating (i.e., most numerous) label, such that

\[
y^e_i = [0, \ldots, 1, \ldots, 0], n_j = \max(n_0, \ldots, n_k),
\]

where \( n_p (p = 0, \ldots, k) \) is the occurrence of the \( p \)-th kind of label observed in \( C_m \). We use \( Y^e_{\text{obs}} \) to denote the label matrix of this category of elector nodes. In practice, we find applying the additional constraint of \( n_j \geq 2 \) can guarantee the reliability of assigned labels.

"Birds of a feather flock together". However, only a small portion (e.g., roughly 1/3 in the Cora dataset) of electors nodes can be directly labeled using Eq. 4. For other electors nodes, we find that the label propagation paradigm leads to unsatisfactory results, mainly due to two reasons: on the one hand, real-world graphs (e.g., Cora, Citeseer, and Pubmed [29]) are often not fully connected, so many nodes cannot receive the broadcast of training labels; on the other hand, long-range dependencies
may bring much noise. As theoretically explained by [8], the stability of Eq. (3) will get strengthened if the number of samplings increases, and $X^c$ will thereupon exhibit stronger class separability (of attributes). This property inspires us to attempt a novel self-learning scheme to handle unlabeled elector nodes. To begin with, we learn a simple binary classifier based on the already-labeled elector nodes (i.e., the union of electors nodes labeled in the last paragraph and those labeled in the previous iterations). Next, after predicting on-the-fly labels using this classifier, we filter out the elector nodes whose labels are assigned with probability lower than a given “labeling threshold”. The above two steps are iteratively performed to produce enough elector nodes with high-quality predicted labels. We use $Y^{c}_{\text{pred}}$ for the corresponding label matrix and that $Y^{c}_{\text{rem}}$ for that of the remaining unlabeled elector nodes. We fill $Y^{c}_{\text{rem}}$ with zeros.

### 3.3 Graph Augmentation and Downstream Coupling

By merging high-quality labeled elector nodes into $G$, we obtain the updated graph $G' = \{V', E', X', Y'\}$, where $V' = V \cup V^c$, $V^c = V^{c}_{\text{obs}} \cup V^{c}_{\text{pred}} \cup V^{c}_{\text{rem}}$ is the set of elector nodes, $X' = X||X^c$, $Y' = Y||Y^c$, $E' = E \cup \{e(v_i,v_j)\}$, $v_i \in V$, $v_j \in V^c$, and $v_i \in C_j$. Apart from the significantly enlarged volume, another outstanding advantage of $G'$ is that, for each dense graph, as elector node becomes the common neighbor of all its voter nodes (i.e., they are linked with new edges), the maximum distance between any two nodes becomes 2, i.e., long-range dependencies get generally shortened. Empirically, we also observe that $G'$ exhibits a relatively uniform distribution of labels and strong class separability of attributes, both of which can facilitate downstream graph modeling.

Finally, $G'$ can be fed into subsequent graph-based SSL models, with the single aggregation operation for $v_i$ at depth $l$ be represented as

$$h^l_i = \sigma\left(\sum_{j \in V_i \cup \{n_i\}} \alpha_{i,j} W h^{l-1}_j\right),$$  

where $h^l_i$ denotes the hidden representation of $v_i$ at the $l$th layer, $V_i$ is the neighbor set of $v_i$, $W$ is a learnable linear transformation matrix, $\sigma(\cdot)$ is an element-wise nonlinear activation function, and $\alpha_{i,j}$ is the evaluation parameter set in feature aggregation (e.g., the attention function of GAT).

Stacking multiple such layers with a task-specific layer yields a Graph Neural Network (GNN) with the standard architecture, which can be directly applied in downstream SSL scenarios.

### 4 Experiments

Following previous studies [11, 26, 29], we demonstrate the effectiveness of ELCO on the widely-adopted semi-supervised node classification benchmark. In practice, the amount of labels in graph data is often orders of magnitude smaller than that of all nodes, i.e., $|Y_{\text{obs}}| \ll |Y - Y_{\text{obs}}|$. To mitigate this issue, graph-based semi-supervised node classification aims to predict the labels of large-scale nodes with a small training set $(V, E, X, Y_{\text{obs}})$. There exist settings for this task, namely transductive learning and inductive learning, which are different in the information visibility. While the former can fully observe and utilize $X$ in both learning and inference stages, the latter is blocked from partial information (i.e., the features of unlabeled vertices) during the learning stage but are fed with the complete dataset during testing. In this paper, we focus on the transductive learning.

#### 4.1 Setup

**Datasets.** Our evaluation is based on three datasets (i.e., Cora, Citeseer, and Pubmed [29, 11, 26]) which are the de facto standards for assessing graph-based SSL algorithms. They are all sampled

| Data    | #Nodes | #Edges | #Features | #Classes | #Train | #Validation | #Test  |
|---------|--------|--------|-----------|----------|--------|-------------|--------|
| Cora    | 2708   | 5429   | 1433      | 7        | 140    | 500         | 1000   |
| Citeseer| 3327   | 4732   | 3703      | 6        | 120    | 500         | 1000   |
| Pubmed  | 19717  | 44388  | 500       | 3        | 60     | 500         | 1000   |


via citation networks, where nodes are for research publications and edges for the citation relation. In Cora and Citeseer, node attributes are represented as bag-of-words, while Pubmed uses TF-IDF weights. For a fair comparison, we adopt the same training/validation/testing splits as Yang et al. [29], Kipf and Welling [11], and Velickovic et al. [26]. Table 1 shows the detailed statistics of the datasets.

Models. To justify the generality of ELCO, we respectively integrate it with GCN and GAT as they are the two most popular graph-based SSL methods in the GNN community. GCN [11], which is proposed by Kipf and Welling, successfully bridges the gap between spectral and spatial methods. Thanks to its scalability, GCN can efficiently learn node representation by encoding adjacency matrix and node attributes. GAT [26], which is developed by Velickovic et al., introduces the multi-head self-attention mechanism to achieve the multi-channel information interaction of adjacent nodes.

As shown in Table 2, we select seven frequently-cited methods and three recently-published approaches as our reference baselines. In addition, we include six and three variants of GCN and GAT, respectively. These selected baselines are not only representative but also very competitive, as some of them claim state-of-the-art performance, such as [24, 27, 31]. The results of all the listed baselines are directly duplicated from the corresponding papers.

Parameters. In practice, we find that our proposed ELCO is robust towards configuration variations. Therefore, we exploit the most straightforward setting for parameters without much fine-tuning. To be specific, the resolution of the overlapping clustering algorithm (i.e., Ego-Splitting [5, 21]) is set at 1.0. In the high-level information diffusion step (cf. § 3.2), we utilize the simple GBDT [7] as our classifier, with a learning rate at 0.25, max depth at 3, and other parameters selected as default. To ensure the quality of generated labels, we set the number of iterative diffusions and the labeling threshold at 10 and 0.99, respectively. During all experiments, we terminate the training when the verification accuracy no longer increases for 2K iterations. Test scores based on the models with the best verification performance are reported.

Table 2: Accuracy (%) of the node classification benchmark. The highest performance per dataset is highlighted in **bold**. The ± error bar denotes the standard deviation in 10 independent trials. ‘-’ means the corresponding value has not been published in the original paper.

| Method            | Cora   | Citeseer | Pubmed |
|-------------------|--------|----------|--------|
| Reference baselines |        |          |        |
| Gaussian Fields   | 68.0   | 45.3     | 63.0   |
| Deep-Semi         | 59.0   | 59.6     | 71.7   |
| Manifold Reg.     | 59.5   | 60.1     | 70.7   |
| Deep-Walk         | 67.2   | 43.2     | 65.3   |
| Link-based        | 75.1   | 69.1     | 73.9   |
| Planetoid         | 75.7   | 64.7     | 74.4   |
| MoNet             | 81.7   | -        | 79.0   |
| SIG-VAE           | 79.7   | 70.4     | 79.3   |
| CurvGN-128        | 82.7±0.7 | 72.1±0.6 | 79.2±0.5 |
| GIL               | 86.2   | 74.1     | 83.1   |
| GCN-based methods |        |          |        |
| Chebyshev         | 81.2   | 69.8     | 74.4   |
| TAGCN             | 83.3   | 72.5     | 79.0   |
| TO-GCN            | 83.1   | 72.7     | 79.5   |
| DGCN             | 83.5   | 72.6     | 80.0   |
| ContiGCN          | 82.0±0.3 | 72.7±0.7 | 79.5±0.5 |
| LSM_GCN           | 82.5±0.2 | 74.4±0.3 | 77.9±0.4 |
| GCN               | 81.5   | 70.3     | 79.0   |
| **ELCO-GCN(Ours)** | 85.6±0.4 | 75.7±0.3 | 83.2±0.4 |
| GAT-based methods |        |          |        |
| LSM_GAT           | 82.9±0.3 | 73.1±0.5 | 77.6±0.7 |
| **ELCO-GAT(Ours)** | 87.6±0.5 | 76.7±0.4 | 83.7±0.4 |

\*1 In particular, GAT_{128}*GAM uses 128 hidden units, which is more than the original GAT (16).
4.2 Main Results

Table 2 reports the results of our baseline methods, base algorithms (GCN and GAT), and ELCO-enhanced models (ELCO-GCN and ELCO-GAT). To reduce randomness, we run each model for 10 independent trials and calculate the average score and standard deviation. On all the three datasets, ELCO-GAT consistently sets new state-of-the-art performance, with margins of 0.6% to 3.3% compared with the best baselines which are not coupled with ELCO. Before being stacked with ELCO, the base GCN is 1.2% inferior to GAT on average; after the data augmentation, ELCO-GCN still falls behind ELCO-GAT. However, compared with other baseline methods, ELCO-GCN ranks second on Citeseer and Pubmed and fourth on Cora, exhibiting strong competitiveness. Please note that, neither GCN nor GAT achieves outstanding accuracies compared with their strong counterparts: more concretely, even the original GAT cannot rank within the top three (with ELCO-enhanced models excluded) on any dataset. This fact emphases the substantial effectiveness of ELCO.

When calculating the specific accuracy enhancement brought by ELCO, we witness very significant 4.1% to 5.4% and 4.2% to 4.7% increases for GCN and GAT, respectively. Put these increments in context: among all the other GCN-based approaches, the range of performance gain over GCN is -4.6% to 4.1%; among all the other GAT-based ones, it is -1.4% to 2.4% over GAT. It is therefore recommended that ELCO be adopted as a standard by graph-based SSL pipelines.

4.3 Further Discussion

In order to obtain more insights for our proposed methods, we perform experiments to investigate the following research questions:

Question 1: Do elector nodes have better attribute class separability than voter nodes?

Question 2: Is $G'$ superior to $G$ in terms of their satisfaction of Criterion $\mathcal{C}$?

To begin with, in Fig. 3 we visualize (original) voter nodes and (generated) elector nodes with attributes as axes. While the former seem more crisscross, the latter exhibit clearer “community borders”, demonstrating that elector nodes have better class separability of attributes. For more rigorous comparisons, we leverage two widely-adopted metrics. Firstly, for voter and elector nodes we measure the Error Rate of Linear Classifier (L2), which is the direct estimation of separability [12] (lower L2 means better separability). As shown in Table 3a on all datasets elector nodes yield significantly-lower L2, indicating their overall separability is much better. In addition, we investigate a task-driven metric, i.e., to see if node attributes can be precisely classified with a simple model. In
Xu et al. [27] propose a graph inference learning framework to model node labels topologically. In which counts the percentage of dominating labels in each cluster partitioned by $E$ with our implementation of $E$ to our knowledge, $E$ Table 3b, we observe an average accuracy gap of 14.5%, meaning that elector nodes are easier to get high-quality nodes, which also exhibit refined class separability of attributes. Results of extensive

5 Related Work

**Generative graph-based SSL models.** Due to the inherent uncertainty of real-world graphs [17], the community has witnessed an increasing interest in analyzing graphs with generative models. For example, Stretcu et al. [24] combine deep learning and label propagation, and utilize extra components to determine the label sharing between nodes. Ma et al. [14] exploit scalable variational inference to approximate the Bayesian posterior of the joint distribution of node features, predictions, and graph structure. Similarly, based on encapsulating attributes, paths, and local graph structures, Xu et al. [27] propose a graph inference learning framework to model node labels topologically. In addition, Ye et al. [30] leverage discrete graph curvature to measure to what extent the neighborhoods of a node pair are structurally related.

**Topological refinement algorithms.** On the other hand, some authors attempt to reduce the impact of the aforesaid uncertainty by adjusting the topology of graph. DropEdge [20] randomly removes edges to prevent excessive smoothing. The model of Abu-El-Haija et al. [1] mixes the features of multi-hop neighbors by short-circuiting distant nodes. Li et al. [28] adjust both inter-community and intra-community edges to optimize the graph topology. Shi et al. [23] maximize consistency for aggregate information by aligning networks at both topological and semantic levels. Jiang et al. [10] incorporate a robust norm feature learning mechanism with graph convolution for SSL with constraints.

To our knowledge, ELCO is the first approach to bridge the gap between graph generative models and topological refinement algorithms, which improves the original graph by adding high-quality nodes generated through joint modeling structure and attribute signals.

6 Conclusion

In this paper, we propose a simple yet effective ELCO framework, which boosts the performance of graph-based SSL models by augmenting training resources. As the first attempt to expand the existing typology, our pipeline can be regarded as a continued graph generation process based on the input information. Aiming to strengthen the label similarity within dense subgraphs, ELCO generates high-quality nodes, which also exhibit refined class separability of attributes. Results of extensive
evaluations indicate that this generic pre-processing technique can dramatically enhance the base algorithms and further outperform state-of-the-art baselines. Followup experiments and analyses present more insights regarding the superiority of ELCO. In the future, we will test ELCO in more setups, as well as explore other graph augmentation strategies.

**Broader Impact**

The ELCO framework significantly boosts the performance of graph-based semi-supervised models, thus benefiting real-world applications which process graph data, e.g., social networks, sensor networks, and molecular structures. Apart from the node classification task which is tested, ELCO may be further applied in a wider spectrum of tasks, including link prediction, data completion, and graph generation.

Apart from a generic augmentation algorithm for graph data, our method can also be regarded as a novel approach to exploiting the existing internal dependencies in low-resource scenarios. Thus, if a specific task from other research fields (e.g., natural language processing, computer vision, data mining, etc.) involves data with characteristics of graph signals, e.g., smoothness/similarity of features within neighboring regions, then ELCO can be possibly adopted.

Please also be aware of some known risks and limitations of our framework. Firstly, when the given graph is too sparse or its internal connections are close to saturation, ELCO may fail to provide satisfactory results, i.e., the augmented data is too few or has low quality. Besides, without explicit mechanisms to handle bias originally introduced by the input, ELCO may yield biased output. Lastly, since the generation of new nodes depends on dense subgraphs, an excessive number of such structures may bring considerable computational overhead.

We would encourage researchers to explore further applications of our method. To mitigate the aforementioned risks and limitations and improve the real-world usability of ELCO, we also welcome all kinds of improvements and enhancements from any research field.

**References**

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrany Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *ICML*, pages 21–29, 2019.

[2] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, 7:2399–2434, 2006.

[3] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3837–3845, 2016.

[4] Jian Du, Shanghang Zhang, Guanhang Wu, José M. F. Moura, and Soummya Kar. Topology adaptive graph convolutional networks. *CoRR*, abs/1710.10370, 2017.

[5] Alessandro Epasto, Silvio Lattanzi, and Renato Paes Leme. Ego-splitting framework: from non-overlapping to overlapping clusters. In *KDD*, pages 145–154, 2017.

[6] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In *NIPS*, pages 6530–6539, 2017.

[7] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[8] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[9] Arman Hasanzadeh, Ehsan Hajarimezanlai, Krishna R. Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. Semi-implicit graph variational auto-encoders. In *NeurIPS*, pages 10711–10722, 2019.

[10] Bo Jiang and Ziyan Zhang. Robustgns: Robust norm graph convolutional networks in the presence of node missing data and large noises. *CoRR*, abs/2003.10130, 2020.

[11] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.
[12] Ana Carolina Lorena, Luís Paulo F. Garcia, Jens Lehmann, Marcílio Carlos Pereira de Souto, and Tin Kam Ho. How complex is your classification problem?: A survey on measuring classification complexity. *ACM Comput. Surv.*, 52(5):107:1–107:34, 2019.

[13] Qing Lu and Lise Getoor. Link-based classification. In *ICML*, pages 496–503, 2003.

[14] Jiaqi Ma, Weijing Tang, Ji Zhu, and Qiaozhu Mei. A flexible generative framework for graph-based semi-supervised learning. In *NeurIPS*, pages 3276–3285, 2019.

[15] Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller, editors. *Neural Networks: Tricks of the Trade - Second Edition*, volume 7700 of *Lecture Notes in Computer Science*. Springer, 2012.

[16] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, pages 5425–5434, 2017.

[17] Berndt Müller, Joachim Reinahardt, and Michael T Strickland. *Neural networks: an introduction*. Springer Science & Business Media, 2012.

[18] Francesco Orsini, Daniele Baracchi, and Paolo Frasconi. Shift aggregate extract networks. *Front. Robotics and AI*, 2018, 2018.

[19] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *KDD*, pages 701–710, 2014.

[20] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *ICLR*, 2020.

[21] Benedek Rozemberczki, Oliver Kiss, and Rik Sarkar. An api oriented open-source python framework for unsupervised learning on graphs, 2020.

[22] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, pages 593–607, 2018.

[23] Min Shi, Yufei Tang, and Xingquan Zhu. Topology and content co-alignment graph convolutional learning. *CoRR*, abs/2003.12806, 2020.

[24] Otilia Stretcu, Krishnamurthy Viswanathan, Dana Movshovitz-Attias, Emmanouil A. Platanios, Sujith Ravi, and Andrew Tomkins. Graph agreement models for semi-supervised learning. In *NeurIPS*, pages 8710–8720, 2019.

[25] Shikhar Vashishth, Prateek Yadav, Manik Bhandari, and Partha P. Talukdar. Confidence-based graph convolutional networks for semi-supervised learning. In *AISTATS*, pages 1792–1801, 2019.

[26] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[27] Chunyan Xu, Zhen Cui, Xiaobin Hong, Tong Zhang, Jian Yang, and Wei Liu. Graph inference learning for semi-supervised classification. In *ICLR*, 2020.

[28] Liang Yang, Zesheng Kang, Xiaochun Cao, Di Jin, Bo Yang, and Yuanfang Guo. Topology optimization based graph convolutional network. In *IJCAI*, pages 4054–4061, 2019.

[29] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, pages 40–48, 2016.

[30] Ze Ye, Kin Sum Liu, Tengfei Ma, Jie Gao, and Chao Chen. Curvature graph network. In *ICLR*, 2020.

[31] Kai Zhang, Yaokang Zhu, Jun Wang, and Jie Zhang. Adaptive structural fingerprints for graph attention networks. In *CILR*, 2020.

[32] Xiaoqin Zhu, Zoubin Ghahramani, and John D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.

[33] Chenyi Zhuang and Qiang Ma. Dual graph convolutional networks for graph-based semi-supervised classification. In *WWW*, pages 499–508, 2018.
A Substructre-based Overlapping Clustering

In this paper, the overlapping clustering algorithm we use to obtain dense graphs is Ego-Splitting [5]. As discussed in § 3.1, it boils into performing non-overlapping clustering in two phases. Compared with the fully-connected graph which the original implementation of Ego-Splitting handles, in our study we make some adaptations as citation networks can be non-fully-connected. The resulting pseudo code is shown in Algorithm 1.

Algorithm 1: Ego-Splitting for citation networks

Data: input graph $G = \{V, E\}$, vertex set $V$, edge set $E$, number of independent connected subgraph in $G$. $N_{sub}$. 

Result: cluster set $S'$.

1. for $i \leftarrow 1$ to $N_{sub}$ do
2.   $V_{sub}^{i}, E_{sub}^{i} \leftarrow G_{sub}^{i}$ ;
3.   for $j \leftarrow 1$ to $|V_{sub}^{i}|$ do
4.     ego – net($v_{i,j}^{sub}$) $\leftarrow$ local clustering $A^{l}(V_{i,j}^{sub}, E_{i,j}^{sub})$ ;
5.     if $v_{i,j}^{sub}$ in $n$ ego – nets then
6.       create $n - 1$ personas in $V^{p}$ ;
7.       split edges of $v_{i,j}^{sub}$ to its personas and form $E^{p}$ ;
8.   end
9. end
10. $S'' \leftarrow$ global clustering $A^{g}(V^{p}, E^{p})$ ;
11. for $j \leftarrow |S''|$ do
12.   $C_{i,j}(C_{i,j}') = \{v \in V_{i,j}^{sub} | \exists k$ s.t. $v_{k} \in C_{i,j}', C_{i,j}' \in S'' \}$ ;
13. end
14. $S' = \{C(C') | C' \in S'' \}$

B Comparing Training Processes

To compare the optimization processes before and after applying the ELCO technique, we record the node classification performance at each epoch when conducting experiments in § 4.
Fig. 5 plots the variation curves of classification accuracy on three test sets during training. When the scores gradually tend to converge (>50 epochs), we observe that the curves of ELCO-GCN and ELCO-GAT are stably above those of GCN and GAT, respectively. This phenomenon demonstrates that the effectiveness of ELCO is robust during the whole training process.

C    Ablation Studies on the Self-learning Module

To see how the composition of nodes fed into the simple classifier (i.e., GBDT) affects the prediction accuracy when generating $Y_{c \text{\_pred}}$ in § 3.2 we conduct ablation studies on various versions of samples: already-labeled elector nodes (i.e., our implemented version), annotated voter nodes in $G$, and the mixture of both. Following our setup in § 4 we set the labeling threshold at 0.99.

As illustrated in Fig. 6 training with voter nodes only cannot even converge (i.e., the binary classification accuracy is always around 0.5 and non-growing). We identify the unsatisfactory attribute class separability as the main cause: it makes training a simple classifier unrealistic. If elector nodes are used for training GBDT, we can see that a large number of high-quality labeled nodes can get harvested through iterative training, thanks to the strong class separability (cf. § 4.3). If we mix two types of nodes, the self-learning process can achieve convergence, but the final prediction accuracy still significantly falls behind our implemented setting, i.e., using elector nodes only.